

# Introduction to HTML and CSS

**HTML**



**CSS**



**SoftUni Team**

**Technical Trainers**



**SoftUni**



**Software University**  
<https://softuni.bg>

# Table of Contents

## 1. What is HTML?

- HTML Tags
- Metadata Section
- Indentation & Code Formatting

## 2. CSS Basic

- CSS Syntax
- Inheritance & Specificity
- CSS Selectors
- CSS Properties



Have a Question?



sli.do

#HTML-CSS



# Introduction to HTML

# What is HTML?

- Markup language for **describing web pages**
- A markup language is **a set of markup tags**
- HTML documents contain **HTML tags** and **plain text**
- The tags **describe document content**



# HTML Tags

- Keywords surrounded by angle brackets
- Normally come in pairs



```
<html>
  <head>
    <title>Simple HTML document example</title>
  </head>
  <body>
    <h1>Simple HTML document example</h1>
  </body>
</html>
```

# Main HTML Tags

- Describes the start and end of the web page/document

```
<html></html>
```

- Describes the start and end of the metadata section of the document

```
<head></head>
```

- Describes the start and end of the visible page content

```
<body></body>
```



# Common HTML Tags

- headings - **h1, h2, h3...**
- paragraphs - **p**
- forms - **form, fieldset, legend...**
- text formatting - **strong, em**
- links - **a**
- images - **img**
- quotes - **cite, blockquote**
- lists - **ol, ul, dl**
- tables - **table, thead, tbody, tr, th, td**



# Tag Attributes

- Tags elements **can have attributes**
- Attributes provide **additional information** about HTML elements
- Attributes are always specified in the start tag
- Attributes come in **name/value pairs**



# Tags Examples

- LINKS

```
<a href="http://initlab.org">init Lab</a>
```

- IMAGES

```

```

- TABLES

```
<table width="100%" cellspacing="0" cellpadding="0">
```



# Metadata Section

- The `<head>` element - a **container** for all the head elements
- Elements inside `<head>` can:
  - **include scripts**
  - instruct the browser where to **find style sheets**
  - provide **meta information**
  - and more
- The following tags can be added to the head section:
  - `<title>`, `<style>`, `<meta>`, `<link>`, `<script>`, `<noscript>`



# Metadata Example

- The <meta> tag provides additional information about the HTML document



```
<!-- Define keywords for search engines: -->
<meta name="keywords" content="HTML, CSS, JavaScript">
<!--Define a description of your web page:-->
<meta name="description" content="Free Web tutorials">
<!--Define the author of a page:-->
<meta name="author" content="Hege Refsnes">
<!--Refresh document every 30 seconds:-->
<meta http-equiv="refresh" content="30">
```

- The <link> tag defines the relationship between a document and an external resource.

```
<head>
  <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

- The <link> tag is most used to link to style sheets

# Indentation Example

- Indentation and code formatting is really important!

```
<form action="#" method="get">
  <fieldset>
    <legend>Login information</legend>

    <label for="username">Username</label>
    <p><input type="text" name="usr" id="usr" /></p>

    <label for="password">Password</label>
    <p><input type="text" name="pass" id="pass" /></p>
  </fieldset>
</form>
```



# CSS Basics

# What is CSS?

- CSS stands for *Cascading Style Sheets*
- Styles define the **visual presentation of HTML elements**
- **Separation** between **semantic content** and **visual presentation** can be achieved
- External visual style guide **shared across all pages** of your site
- Change the style guide - change all pages **visual presentation**



# CSS Syntax

- Every CSS document is **a collection of CSS rules**
- CSS rule has two main parts:
  - Selector
  - One or more declarations

```
[selector] {  
  [declaration]  
  [declaration]  
}
```

- Each declaration consists of a property and a value



# CSS Selectors

- The selector is a identifier of the HTML element or the group of HTML elements you want to style

```
body {  
...  
}
```

# CSS Declarations

- Declaration groups are surrounded by **curly brackets**
- Declarations **end with a semicolon**

```
{  
  font: 16px/1.5 Verdana, sans-serif;  
  color: #333;  
}
```

# Element Selectors

- Using the HTML tag **names as selectors** will apply styles to **all tags** in the document
- Styling all titles of level 1:

```
h1 {  
    text-align: center;  
    color: #000;  
}
```

- Adding more white space after each paragraph:

```
p {  
    padding-bottom: 15px;  
}
```



# The id Selector

- The id selector uses the id attribute of the HTML element, and is defined with a "#"
- Using the id selector will give you the **exact element** you are referring to



```
#header {  
    border: 1px solid #CCC;  
    border-width: 1px 0;  
}
```

# The class Selector

- Uses the HTML class attribute, and is defined with a ". ".
- Allows you to set a particular style for many elements

```
.mod {  
    border-top: 1px solid #000;  
}
```

- You can also specify that only one HTML tag should be affected by a class.

```
p.right {  
    text-align: right;  
}
```



# Pseudo-classes

- CSS pseudo-classes are used to add special effects to some selectors

```
selector:pseudo-class {property:value;}
```

- Changing the styles of a link when the user's mouse is over it

```
a:hover {  
    text-decoration: underline;  
    color: #C00;  
}
```



# Combining Selectors



```
<h1 id="header" class="intro">HTML and CSS</h1>
```

```
h1#header.intro {  
    color: #F00;  
    border: 2px solidcurrentColor;  
}
```

# Adding CSS To Our Html Documents

- There are three ways of inserting a style sheet:
  - External style sheet
  - Internal style sheet
  - Inline style



# External Style Sheet

- Ideal when the style is applied to many pages
- You can change the look of an entire Web site by changing one file
- Each page must link to the style sheet using the tag. The tag goes inside the head section:

```
<head>
  <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

# Internal Style Sheet

- Should be used when a single document has a unique style
- You define **internal** styles in the head section of an HTML page, by using the inline styles
  - Loses many of the advantages of style sheets by **mixing content** with presentation
  - **Use this method sparingly!**



# Inheritance & Specificity

- CSS relies heavily on specificity and style overwriting
- In increasing order of priority.
  - External <link>
  - In the <head>
  - Inline style attribute
  - Using !important



# SELECTOR PRIORITY (SPECIFICITY)

0

0

0

0

inline styles

# of id selectors

# of class selectors

# of element selectors

```
p { color: #FFF; }
```

```
0, 0, 0, 1
```

```
.intro { color: #345678; }
```

```
0, 0, 1, 0
```

```
#header { color: #000; }
```

```
0, 1, 0, 0
```

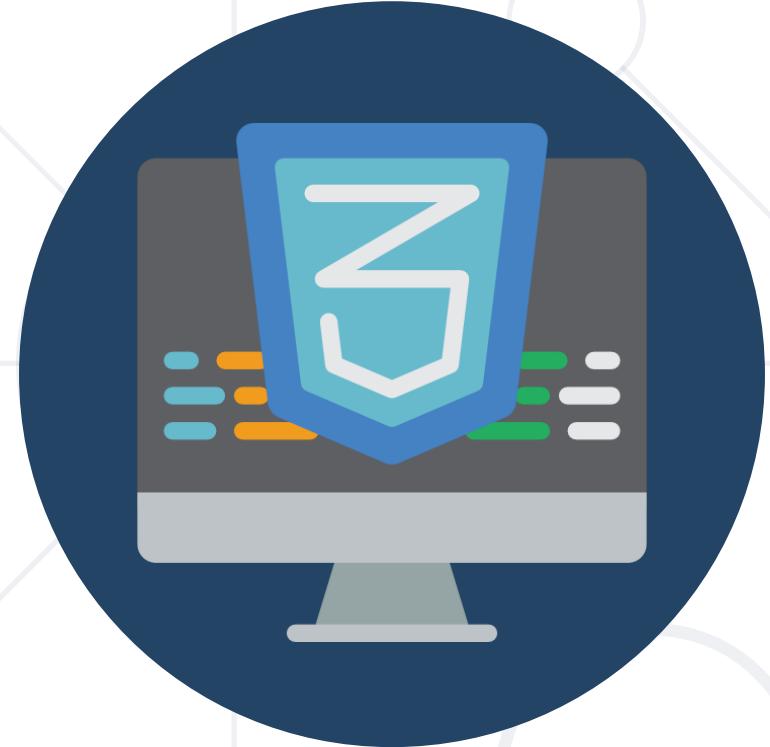
```
<p style="color: #000;">Text</p>
```

```
1, 0, 0, 0
```

```
p { color: #000 !important; }
```

God-mode





# CSS Properties

# Background

- background-color: #fff;
- background-image: url(..../image.png);
- background-repeat: repeat | repeat-x | repeat-y | no-repeat;
- background-position: top left;



# Text

- color: #000;
- text-align: left | right | center | justify;
- text-decoration: underline | overline | line-through | none;
- text-transform: uppercase | lowercase | capitalize;
- text-indent: 50px;



# Font

- font-family: Verdana, Arial, sans-serif;
- font-style: italic | normal;
- font-size: 16px;
- font-weight: bold | normal;



# Border

- border-style: solid | dotted | dashed | double;
- border-color: #C00;
- border-width: 2px;



# Margin

- margin-top: 10px;
- margin-right: 10px;
- margin-bottom: 10px;
- margin-left: 10px;



# Padding

- padding-top: 10px;
- padding-right: 10px;
- padding-bottom: 10px;
- padding-left: 10px;



# Lists Styles

- `list-style-type: none | circle | disc;`
- `list-style-position: inside | outside;`
- `list-style-image: url('../image');`





# Live Exercises

- HTML - Markup language for **describing web pages**
- CSS **defines the visual presentation** of HTML elements
- CSS **relies heavily on specificity and style overwriting**



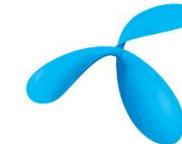
# SoftUni Diamond Partners



**xssoftware**



**SBTech**  
*we know sports*



**telenor**



**SoftwareGroup**  
*doing it right*

**NETPEAK**



**SmartIT**



**Postbank**

*Решения за твоето упре*



**INDEAVR**  
*Serving the high achievers*

 **INFRASTICS®**



**STEMO®**

*Computer Systems & Software*

**SUPERHOSTING.BG**

# SoftUni Organizational Partners



ИНФОРМАЦИОННО  
ОБСЛУЖВАНЕ

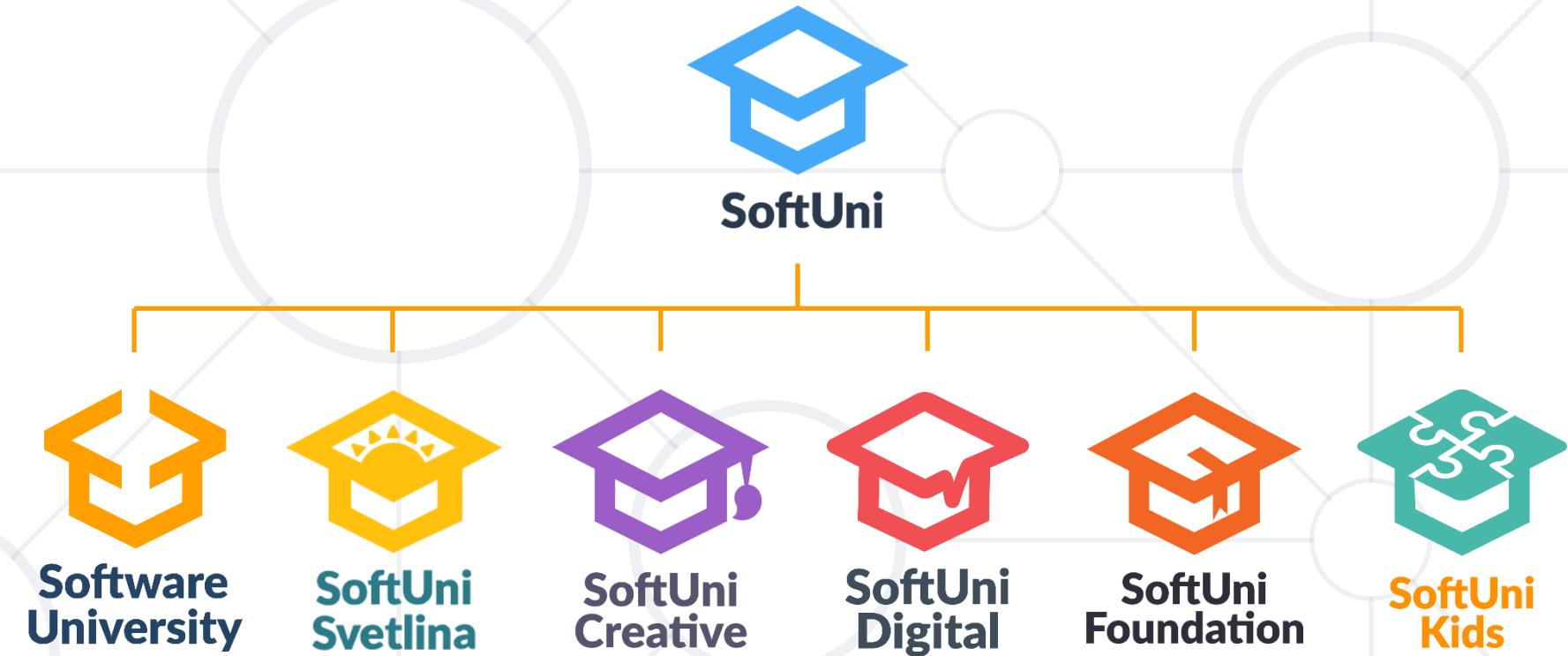
OneBit  
SOFTWARE



Lukanet.com



# Questions?



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>



# Trainings @ Software University (SoftUni)



- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Software University Forums
  - [forum.softuni.bg](http://forum.softuni.bg)



Software  
University



# HTML STRUCTURE



SoftUni Team  
Technical Trainers



SoftUni



Software University  
<https://softuni.bg>

# Table of Contents

1. Semantic Markup
2. HTML Tags
3. Forms
4. Tables
5. Microdata



Have a Question?

**sli.do**

**#HTML-CSS**



# Semantic HTML

# What is Semantic HTML?

- HTML that **introduces meaning** to the web page rather than just presentation

```
<p>Some random text...</p>
```

Indicates that the enclosed text is a paragraph

- This is both semantic and presentational
  - People know what paragraphs are and browsers know how to display them

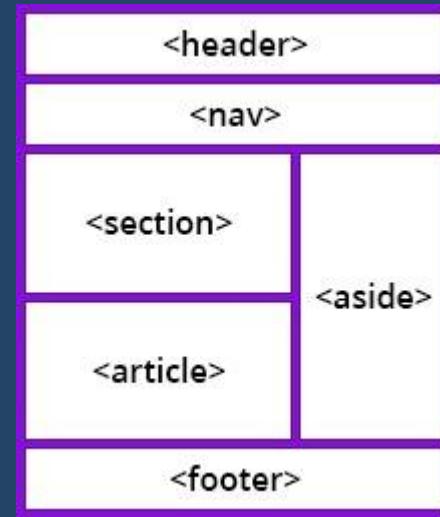


# Why You Should Care About Semantics?

- Provides an **additional information** about that document, which **aids in communication**
- Semantic tags make it **clear** to the **browser** what the **meaning** of a page and its **content** is
  - That clarity is also **communicated** with search engines



# Semantic HTML Tags



# <header></header>

- Represents **introductory content**
- Typically a group of **introductory** or **navigational aids**
- It may contain some heading elements but also other elements like:
  - a logo
  - a search form
  - an author name



# <nav></nav>

- Represents a **section of a page**
- Its purpose is to **provide navigation links**, either within the current document or to other documents
- Common examples of **navigation sections** are
  - menus
  - tables of contents
  - indexes



# <main></main>

- Represents the **dominant content** of the 'body' of a document
- The main content area consists of content that is **directly related** to or **expands** upon:
  - the central topic of a document
  - the central functionality of an application



# <aside></aside>

- Represents a **portion** of a document whose content is only **indirectly related** to the document's **main** content
- Asides are frequently presented as **sidebars** or **call-out boxes**



# <footer></footer>

- Represents a **footer** for its **nearest sectioning content** or **sectioning root element**
- A footer typically contains information about:
  - the **author** of the section
  - the **copyright** data
  - the **links** to related documents



# <article></article>

- Represents a **self-contained composition** in a document, page, application, or site
- Intended to be **independently distributable** or **reusable**
- Examples include:
  - a forum post
  - a magazine
  - newspaper article
  - a blog entry



# <section></section>

- Represents a **standalone section** - which **doesn't have** a more **specific semantic element** to represent it
- Typically, but not always, sections have a heading



# <figure></figure>

- Represents self-contained content
- Frequently with a caption '**figcaption**'
- Typically referenced as a single unit



# <mark></mark>

- Represents text which is **marked** or **highlighted** for reference or notation purposes
- Due to the marked passage's relevance or importance in the enclosing context





# Forms

# Form

- The HTML **form** element represents a document section that contains interactive controls for submitting information to a web server
- A form takes input from the site visitor and posts it to the back-end



# Form Attributes

- **Action** - used to specify where the form data is to be sent to the server after submission of the form
- **Method** - The HTTP method that the browser uses to submit the form
  - POST – Corresponds to the **HTTP POST** method. Form data are included in the body of the form and sent to the server
  - GET – Corresponds to the **HTTP GET** method. Appends form-data into the URL in name/value pairs

# Form Attributes

## ■ Form - Example

```
<form action="/action_page.php" method="get">
  <label for="male">Male</label>
  <input type="radio" name="gender" id="male" value="male">
  <label for="female">Female</label>
  <input type="radio" name="gender" id="female" value="female">
</form>
```

# Form Elements - Input

- <input> element is the most important form element. It can be displayed in several ways, depending on the **type** attribute
  - <input type="text">
  - <input type="number">
  - <input type="password">
  - <input type="email">
  - <input type="search">

# Form Elements - Input

- `<input type="checkbox">`
- `<input type="radio">`
- `<input type="range">`
- `<input type="submit">`
- `<input type="button">`
- `<input type="file">`

# Form Elements - Input

- Input attributes
  - **Value** – specifies the initial value for an input field
  - **Name** – specifies the name of the input element
  - **Placeholder** – specifies a hint that describes the expected value of the input field
  - **Required** – the field must be filled out before submitting the form
  - **Autofocus** – the input should automatically get focus when the page load
  - **Disabled** – specifies that the input field is disabled
  - **Min and Max** – specify the minimum and maximum values

# Form Elements - label

- <label> - defines a label for the others forms elements
- The **for** attribute should be equal to the **id** attribute of the related element to bind them together

```
<form action="/action_page.php" method="get">
  <label for="male">Male</label>
  <input type="radio" name="gender" id="male" value="male">
</form>
```

# Form Elements - fieldset

- **<fieldset>** - used to group related data in a form
- **<legend>** - defines a caption for the **<fieldset>** element

```
<form action="/action_page.php">
  <fieldset>
    <legend>Personal information:</legend>
    First name:<br>
    <input type="text" name="firstname" value="Mickey"><br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

# Form Elements - select

- <select> - defines a drop-down list
- <option> - defines an option that can be selected

```
<select name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

# Form Elements - textarea

- <textarea> - defines a multi-line input field
- Textarea attributes:
  - Rows – specifies the visible number of lines in a text area
  - Cols – specifies the visible width of a text area

```
<textarea name="message" rows="10" cols="30">  
    The cat was playing in the garden.  
</textarea>
```

# Form Elements

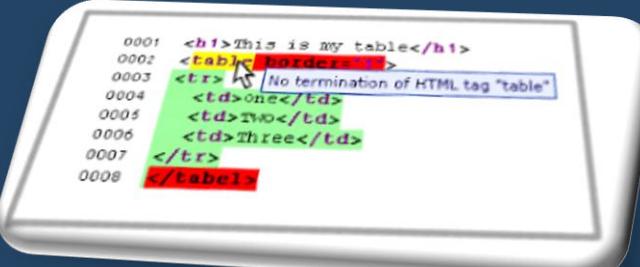
- <button> - defines a clickable button

```
<button type="button">Click Me!</button>
```

- <datalist> - specifies a list of pre-defined options for an <input> element

```
<form action="/action_page.php">
  <input list="browsers">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
  </datalist>
</form>
```

# Tables



A screenshot of a code editor displaying an HTML file with syntax highlighting. The code is as follows:

```
0001 <h1>This is my table</h1>
0002 <table border="1">
0003 <tr> No termination of HTML tag "table"
0004 <td>one</td>
0005 <td>Two</td>
0006 <td>Three</td>
0007 </tr>
0008 </table>
```

The line containing the error is highlighted in red, and a tooltip above it reads "No termination of HTML tag 'table'".

# Table Elements

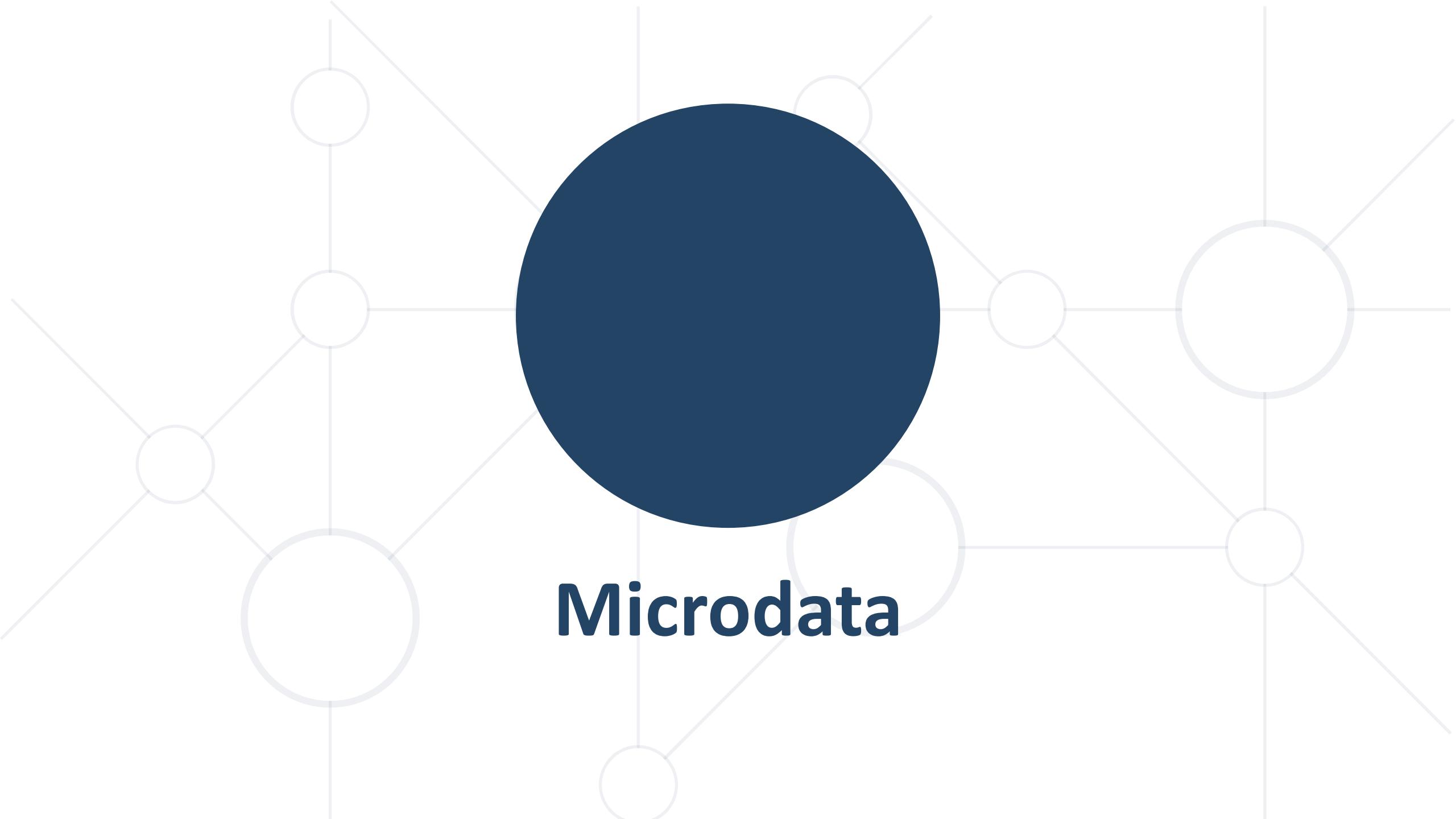
- An HTML table is defined with the `<table>` tag
- `<thead>` tag is used to group header content in the table
- `<tbody>` tag is used to group the body content
- `<tfoot>` tag defines a set of rows summarizing the columns of the table
- `<tr>` tag defines a table row
- `<th>` tag defines a table header
- `<td>` tag defines a table data/cell

# Table Example

```
<table>
  <thead>
    <tr>
      <th>Items</th>
      <th>Expenditure</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>Donuts</th>
      <td>3,000</td>
    </tr>
```

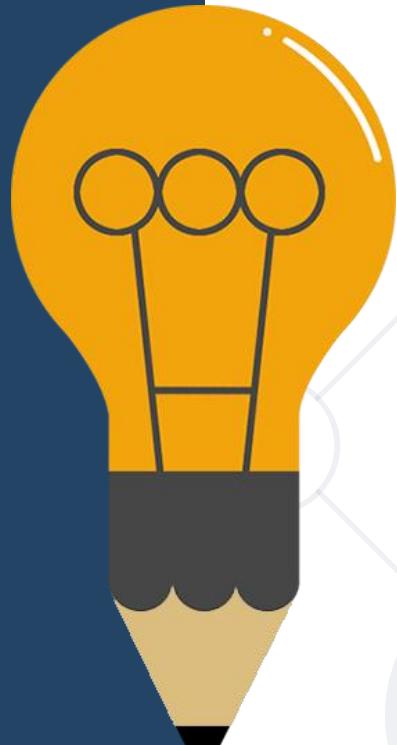
```
    <tr>
      <th>Stationery</th>
      <td>18,000</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <th>Totals</th>
      <td>21,000</td>
    </tr>
  </tfoot>
</table>
```

# Microdata



# What is Microdata?

- Search engines, web crawlers, and browsers can **extract** and **process** Microdata from a web page
- They use it to provide a **richer browsing experience** for users
- Search engines benefit greatly from **direct access** to this structured data
- It allows them to **understand** the **information** on web pages and **provide** more **relevant results** to users



# Microdata - Example



```
<section>
```

Hello, my name is John Doe,  
I am a trainer at SoftUni.  
My friends call me Johnny.  
You can visit my homepage at  
<https://doe.me>.  
I live at 123123, Kamelia str, Sofia.

```
</section>
```

# Microdata - Example (2)

```
<section itemscope itemtype="http://schema.org/Person">  
    Hello, my name is <span itemprop="name">John Doe</span>  
    I am a <span itemprop="jobTitle">trainer</span>.  
    My friends call me <span itemprop="additionalName">Koko</span>.  
    Visit my homepage at <a href="https://doe.me" itemprop="url">https://doe.me</a>.  
<section itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">  
    I live at  
    <span itemprop="streetAddress">123123</span>,  
    <span itemprop="addressLocality">Kamelia str</span>,  
    <span itemprop="addressRegion">Sofia</span>.  
</section>  
</section>
```



# Live Exercises

# Summary

- Semantic HTML
- Tags
- Forms
- Tables
- Microdata



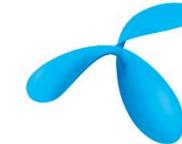
# SoftUni Diamond Partners



**xssoftware**



**SBTech**  
*we know sports*



**telenor**



**SoftwareGroup**  
*doing it right*

**NETPEAK**



**SmartIT**



**Postbank**

*Решения за твоето упре*



**INDEAVR**  
*Serving the high achievers*

 **INFRASTICS®**



**STEMO®**

*Computer Systems & Software*

**SUPERHOSTING.BG**

# SoftUni Organizational Partners



ИНФОРМАЦИОННО  
ОБСЛУЖВАНЕ

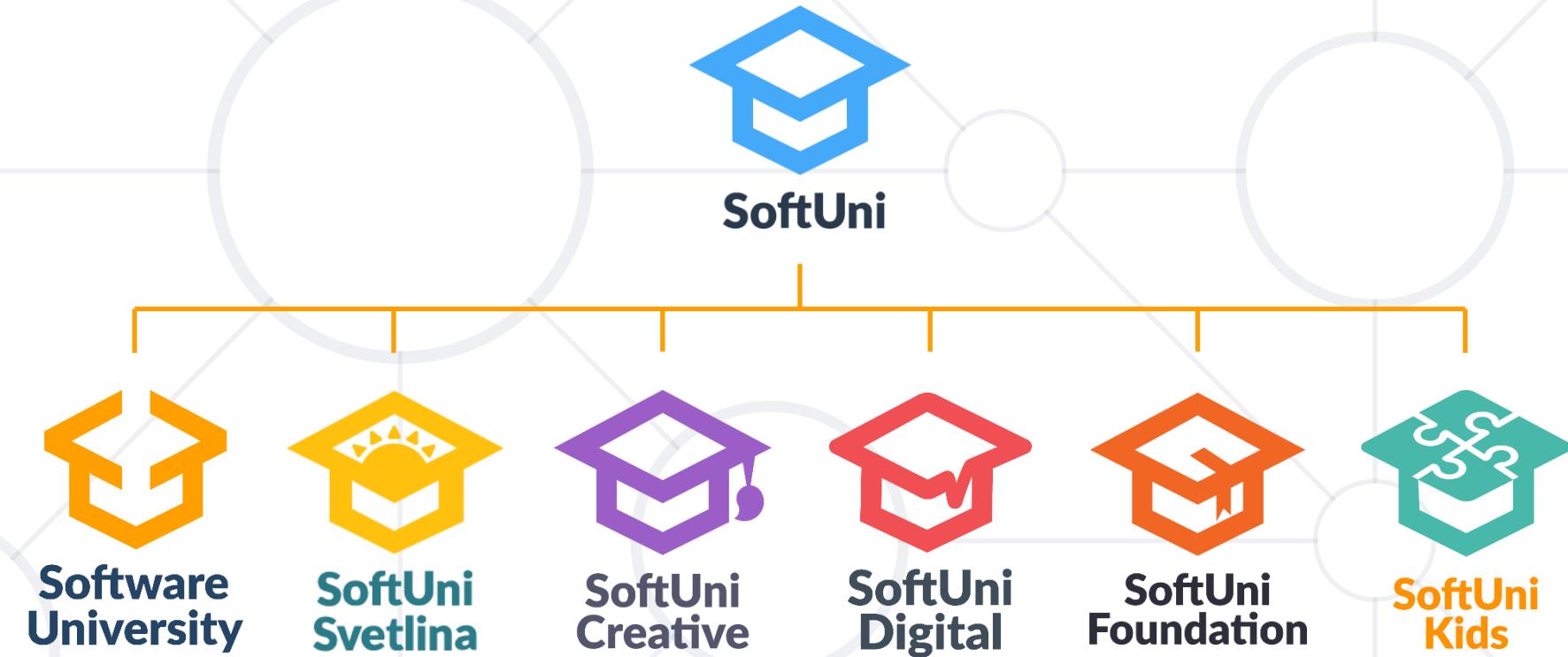
OneBit  
SOFTWARE



Lukanet.com



# Questions?



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>



# Trainings @ Software University (SoftUni)



- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Software University Forums
  - [forum.softuni.bg](http://forum.softuni.bg)



Software  
University



# CSS & TYPOGRAPHY



SoftUni Team  
Technical Trainers



SoftUni



Software University  
<https://softuni.bg>

# Table of Contents

1. Typography
2. Principles Of Readability
3. CSS Properties
4. Font Awesome Icons



Have a Question?



sli.do

#HTML-CSS

# Typography



Typography

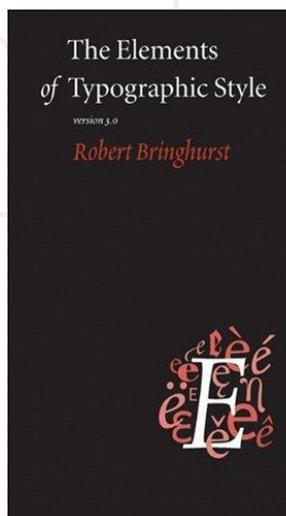
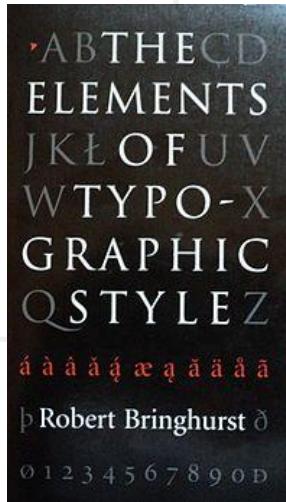
# What is Typography?

- Typography is the art and technique of arranging type to make written language **legible, readable** and **appealing** when displayed
- Typography is the visual component of the written word
- Style or appearance of text
- The art of working with text



# Books

- *The Elements of Typographic Style* – the authoritative book on typography and style
- *The Elements of Typographic Style Applied to the Web* – a practical guide to web typography



# Principles Of Readability

- **Readability** is one of the more important aspects of Web design usability
- Website readability is a measure of how easy it is for visitors to **read** and **understand** online text
- Readability depends on both a text's presentation and its context
- Poor readability scares readers away from the content

# Principles Of Readability

- **Hierarchy** – defines how to read through content
- **Contrast** – the core factor in whether or not text is easy to read
- **Line Height** – the space between individual lines of text
- **Letter Spacing** – the space between each letter in words
- **Line Length** – the number of words per line

# Keys to Readable Typography

- User-Friendly Headers
- Scannable Text
- White Space
- Consistency
- Density of Text
- Emphasis of Important Elements
- Organization of Information
- Good Margins



# CSS Properties

# Font Family

- The **font-family** property specifies the font for an element
- Each font-family property lists **one or more** font families, separated by **commas**

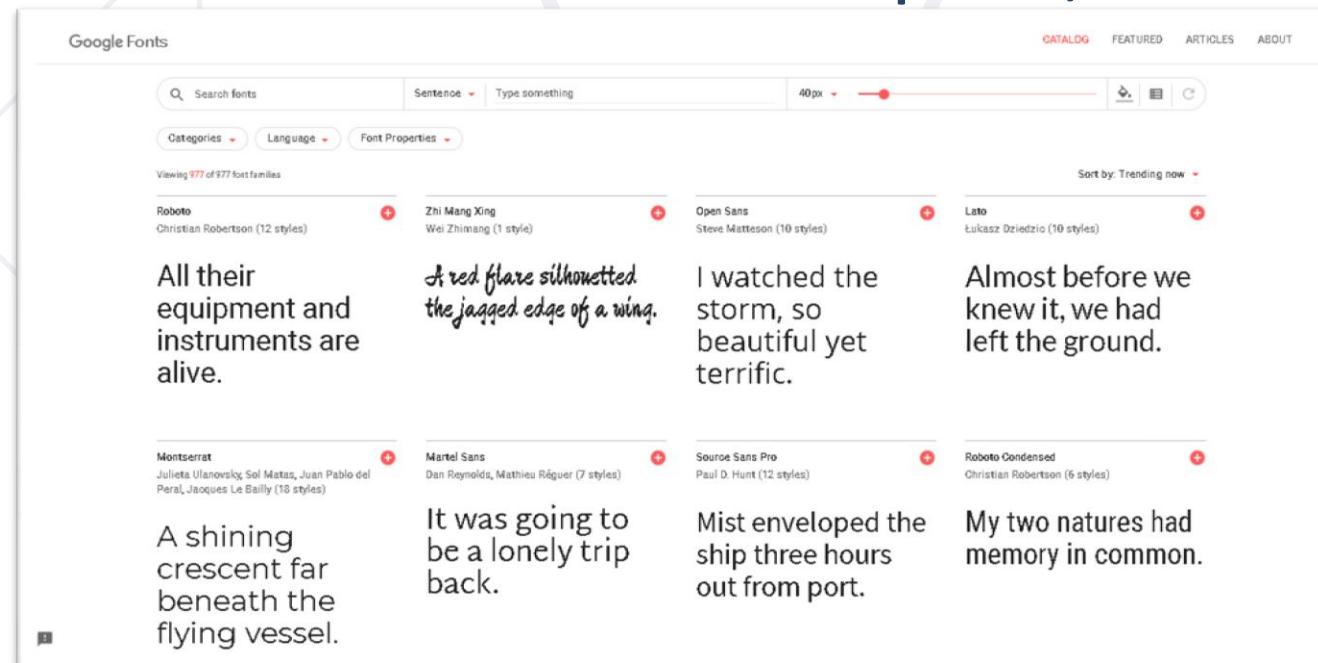
```
body {  
    font-family: <family-name>, <generic-name>;  
}
```

- If the browser does **not support** the first font, it tries the next font

```
body {  
    font-family: Arial, Helvetica, sans-serif;  
}
```

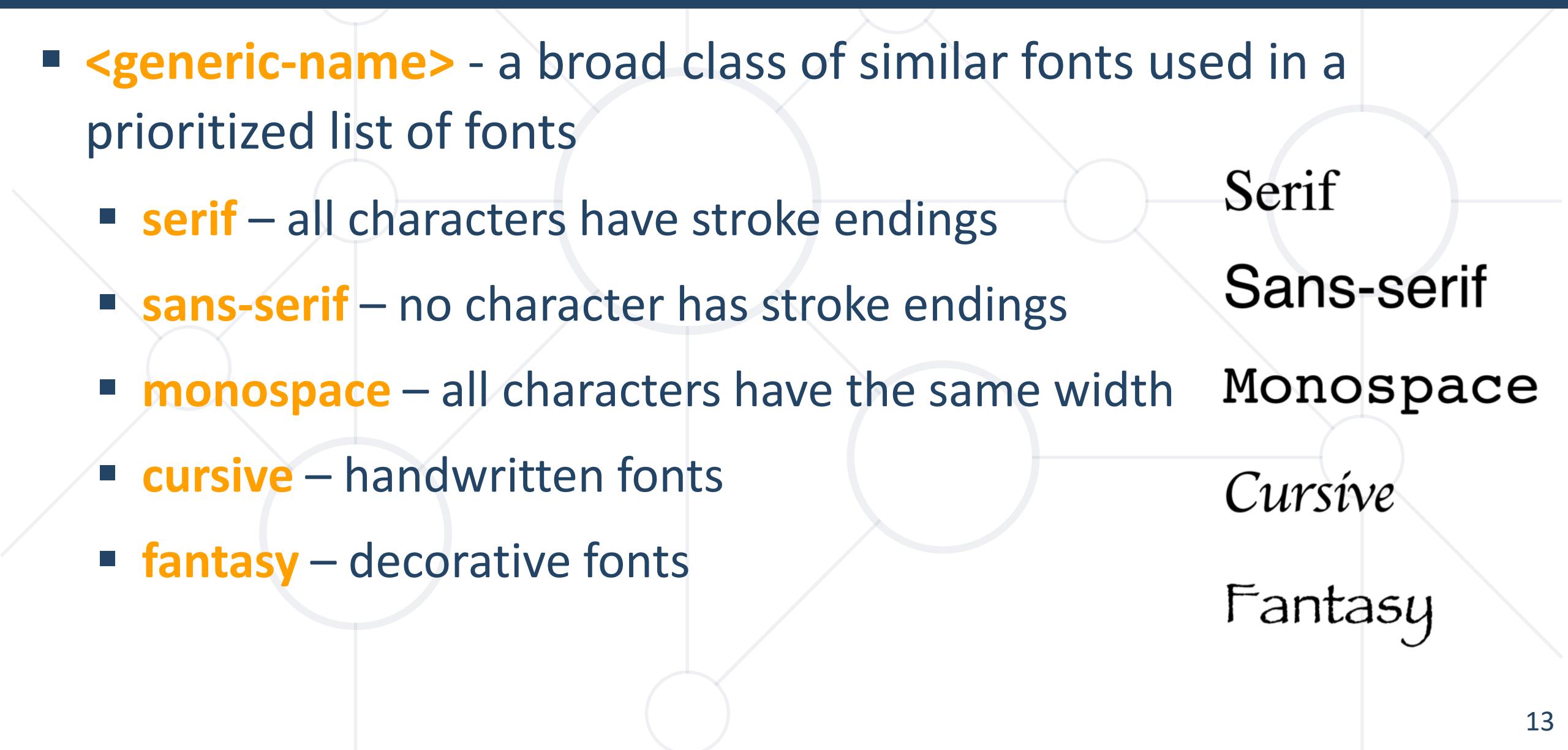
# Family Name

- <family-name> – the name of a font-family, like "Times" and "Helvetica"
- If a font name contains white-space, it must be quoted



<https://fonts.google.com/>

# Generic Name

- **<generic-name>** - a broad class of similar fonts used in a prioritized list of fonts
    - **serif** – all characters have stroke endings
    - **sans-serif** – no character has stroke endings
    - **monospace** – all characters have the same width
    - **cursive** – handwritten fonts
    - **fantasy** – decorative fonts
- 
- The diagram illustrates the classification of generic fonts into five distinct categories, each represented by a circle connected to the main definition of <generic-name>. The categories are: **Serif**, **Sans-serif**, **Monospace**, **Cursive**, and **Fantasy**.

# Font Style

- **Font-style** – defines how much the text is slanted

- normal – the text is not slanted

```
p {  
    font-style: normal;  
}
```

- italic – the letters are slightly slanted

```
p {  
    font-style: italic;  
}
```

- oblique – the letters are more slanted than italic

```
p {  
    font-style: oblique;  
}
```

Normal font style  
*Italic font style*  
*Oblique font style*

# Font Size

- **Font-size** – defines the text size

- Pixel values

```
p {  
    font-size: 16px;  
}
```

- Em values – the value is relative to the parent's font-size

```
p {  
    font-size: 1.2em;  
}
```

Parent container: 18px

Font-size: 1.2em = 21.6px

Font-size: 1.2em =  
25.92px

Font-size:  
1.2em =  
31.104px

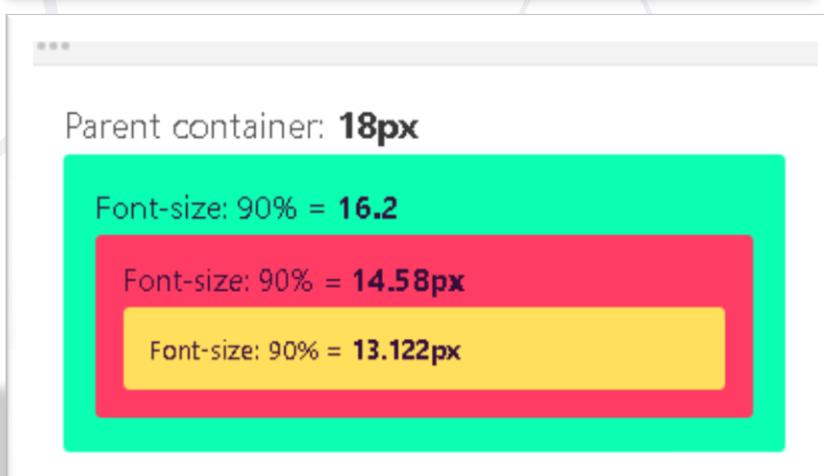
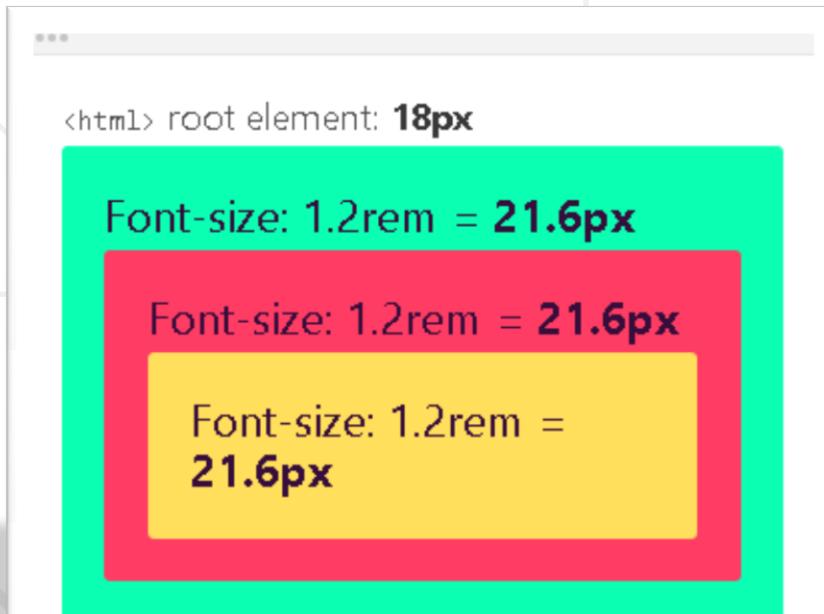
# Font Size

- Rem values – the value is relative to the root element's font-size, which is the `<html>` element

```
p {  
    font-size: 1.2rem;  
}
```

- Percentage values – they act like em values

```
p {  
    font-size: 90%;  
}
```



# Font Weight

- **Font-weight** – defines the weight of the text

- **normal** – the text is in normal weight

```
p {  
    font-weight: normal;  
}
```

Hello world

- **bold** – the text becomes bold

```
p {  
    font-weight: bold;  
}
```

Hello world

# Font Weight

- numeric values

```
p {  
    font-weight: 500;  
}
```

- They all correspond to a particular named weight:

100 – Thin

200 – Extra Light

300 – Light

400 – Normal

500 – Medium

600 – Semi Bold

700 - Bold

800 – Extra Bold

900 – Ultra Bold

# Line Height

- **Line-height** – defines the height of a single line of text
  - Default value – reverts to the default value of the browser

```
p {  
    line-height: normal;  
}
```

...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet.

- Unitless values – the line height will be relative to the font size

```
p {  
    line-height: 1.6;  
}
```

...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet.

# Letter Spacing

- **Letter-spacing** – defines the spacing between the characters of a block of text
  - Normal – the spacing between the characters is normal

```
p {  
  letter-spacing: normal;  
}
```

The quick brown fox jumps over the lazy dog

- Using pixel or em values

```
p {  
  letter-spacing: 2px;  
}
```

The quick brown fox jumps over the lazy dog

```
p {  
  letter-spacing: 0.1em;  
}
```

The quick brown fox jumps over the lazy dog

# Text align

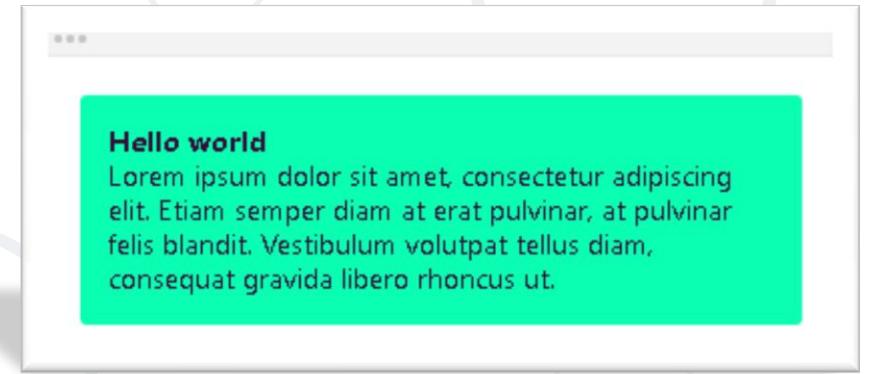
- **Text-align** – defines how the content of the element is horizontally aligned

- Left

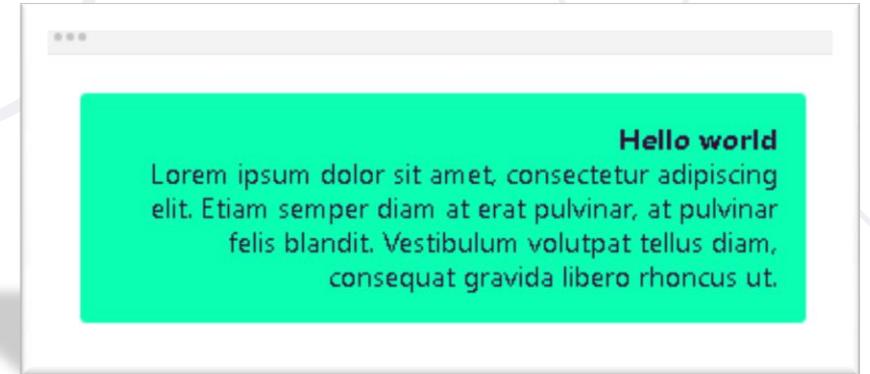
```
p {  
    text-align: left;  
}
```

- Right

```
p {  
    text-align: right;  
}
```



Hello world  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.



Hello world  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

# Text align

## ■ Center

```
p {  
    text-align: center;  
}
```

**Hello world**  
Lorem ipsum dolor sit amet, consectetur adipiscing  
elit. Etiam semper diam at erat pulvinar, at pulvinar  
felis blandit. Vestibulum volutpat tellus diam,  
consequat gravida libero rhoncus ut.

## ■ Justify

```
p {  
    text-align: justify;  
}
```

**Hello world**  
Lorem ipsum dolor sit amet, consectetur adipiscing  
elit. Etiam semper diam at erat pulvinar, at pulvinar  
felis blandit. Vestibulum volutpat tellus diam,  
consequat gravida libero rhoncus ut.

# Text decoration

- **Text-decoration** – defines how the text content of the element is decorated
  - None – removes any text decoration

```
p {  
    text-decoration: none;  
}
```



- Underline – underlines the text content

```
p {  
    text-decoration: underline;  
}
```



# Text indent

- **Text-indent** – defines the indentation of the element's first line of text
  - The text is not indented

```
p {  
    text-indent: 0;  
}
```

...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet.

- The text is indented

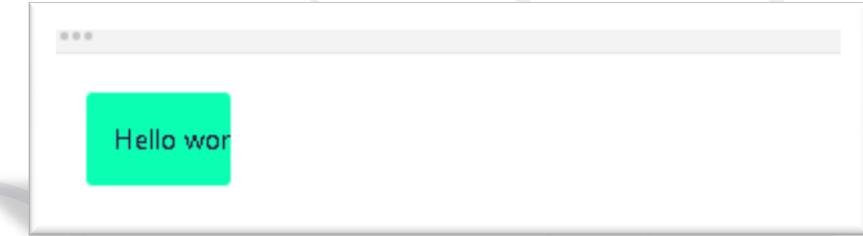
```
p {  
    text-indent: 40px;  
}
```

...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet.

# Text overflow

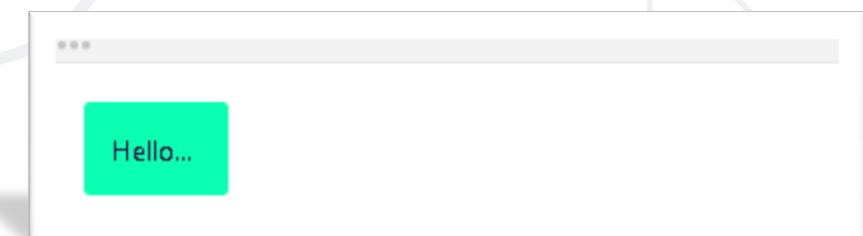
- **Text-overflow** – defines how the hidden text content behaves if it's overflowing
  - Clip – the text content is clipped and not accessible

```
p {  
    text-overflow: clip;  
}
```



- Ellipsis – the overflowing content is replaced by ...

```
p {  
    text-overflow: ellipsis;  
}
```



# Text transform

- **Text-transform** – defines how the text content should be transformed

- None – default value

```
p {  
    text-transform: none;  
}
```

...  
**Hello world!**  
Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Etiam semper diam at erat pulvinar, at pulvinar felis  
blandit. Vestibulum volutpat tellus diam, consequat  
gravida libero rhoncus ut. Maecenas imperdiet felis nisi,  
fringilla luctus felis hendrerit sit amet.

- Capitalize – turns the **first letter** of each word into a capital letter

```
p {  
    text-transform: capitalize;  
}
```

...  
**Hello World!**  
Lorem Ipsum Dolor Sit Amet, Consectetur Adipiscing Elit.  
Etiam Semper Diam At Erat Pulvinar, At Pulvinar Felis  
Blandit. Vestibulum Volutpat Tellus Diam, Consequat  
Gravida Libero Rhoncus Ut. Maecenas Imperdiet Felis Nisi,  
Fringilla Luctus Felis Hendrerit Sit Amet.

# Text transform

- Uppercase – turns all letters into **capital** letters

```
p {  
    text-transform: uppercase;  
}
```

## HELLO WORLD!

LOREM IPSUM DOLOR SIT AMET, CONSETETUR  
ADIPISCING ELIT. ETIAM SEMPER DIAM AT ERAT  
PULVINAR, AT PULVINAR FELIS BLANDIT. VESTIBULUM  
VOLUTPAT TELLUS DIAM, CONSEQUAT GRAVIDA LIBERO  
RHONCUS UT. MAECENAS IMPERDIET FELIS NISI,  
FRINGILLA LUCTUS FELIS HENDRERIT SIT AMET.

- Lowercase – turns all letters into **lowercase** letters

```
p {  
    text-transform: lowercase;  
}
```

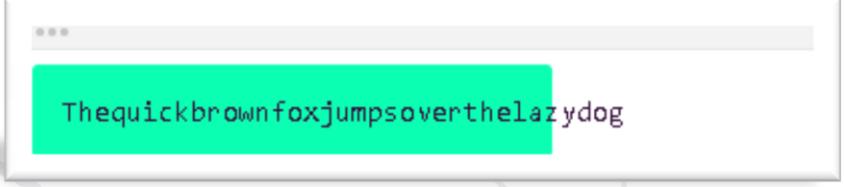
## hello world!

lorem ipsum dolor sit amet, consectetur adipiscing elit.  
etiam semper diam at erat pulvinar, at pulvinar felis  
blandit. vestibulum volutpat tellus diam, consequat gravida  
libero rhoncus ut. maecenas imperdiets felis nisi, fringilla  
luctus felis hendrerit sit amet.

# Word break

- **Word break** – defines how words should break when reaching the end of line
  - Normal – words with no space will **not** break

```
p {  
    word-break: normal;  
}
```



Thequickbrownfoxjumpsoverthelazydog

- Break-all – words with no space will **break** as soon as they reach the end of a line

```
p {  
    word-break: break-all;  
}
```



Thequickbrownfoxjumpsoverthe  
lazydog

# Text shadow

- **Text-shadow** – defines the shadow of the text content
  - None – the text content has **no shadow**

```
p {  
    text-shadow: none;  
}
```

...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Etiam semper diam at erat pulvinar, at pulvinar felis  
blandit. Vestibulum volutpat tellus diam, consequat  
gravida libero rhoncus ut. Maecenas imperdiet felis nisi,  
fringilla luctus felis hendrerit sit amet.

- Text-shadow: <horizontal> <vertical> <blur> <color>

```
p {  
    text-shadow: 2px 4px 10px red;  
}
```

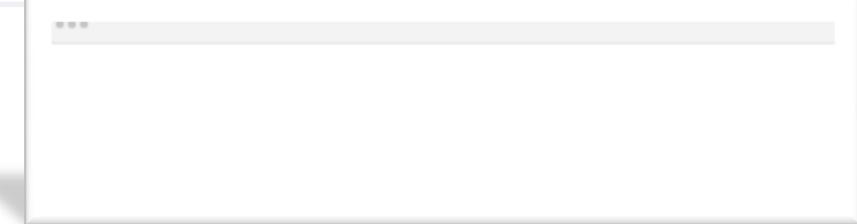
...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Etiam semper diam at erat pulvinar, at pulvinar felis  
blandit. Vestibulum volutpat tellus diam, consequat  
gravida libero rhoncus ut. Maecenas imperdiet felis nisi,  
fringilla luctus felis hendrerit sit amet.

# Text color

- **Color** – defines the color of the text

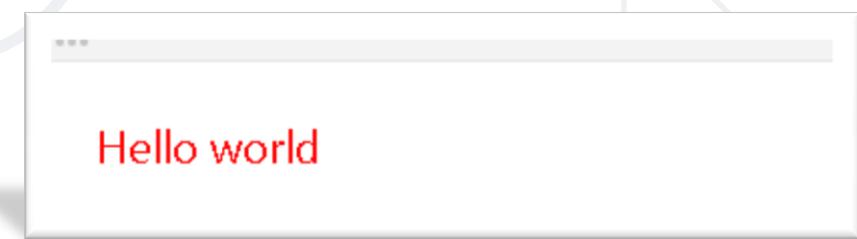
- Transparent

```
p {  
    color: transparent;  
}
```



- You can use one of the 140+ color names

```
p {  
    color: red;  
}
```



# Text color

- You can use **hexadecimal** color codes

```
p {  
    color: #05ffb0;  
}
```

Hello world

- You can use **rgb()** color codes

```
p {  
    color: rgb(50, 115, 220);  
}
```

Hello world

- You can use **rgba()** color codes

```
p {  
    color: rgba(0, 0, 0, 0.5);  
}
```

Hello world

# Background color

- **Background-color** – defines the color of the element's background
  - Transparent

```
p {  
    background-color: transparent;  
}
```



- Color

```
p {  
    background-color: red;  
}
```



- **Cursor** – Sets the mouse cursor when hovering the element

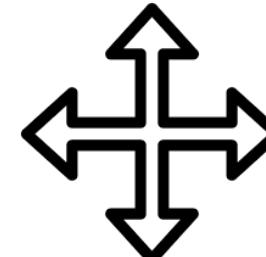
- Pointer

```
p {  
  cursor: pointer  
}
```



- Move

```
p {  
  cursor: move;  
}
```



## ■ Outline

- Outline-width - defines the **width** of the element's outlines
- Outline-style - defines the **style** of the element's outlines
- Outline-color - defines the **color** of the element's outlines

```
p {  
    outline: 4px dotted red;  
}
```





**Icons**  
**Font Awesome**

# Form Awesome

- To use the Font Awesome icons, add the following line inside the `<head>` section of your **HTML** file:

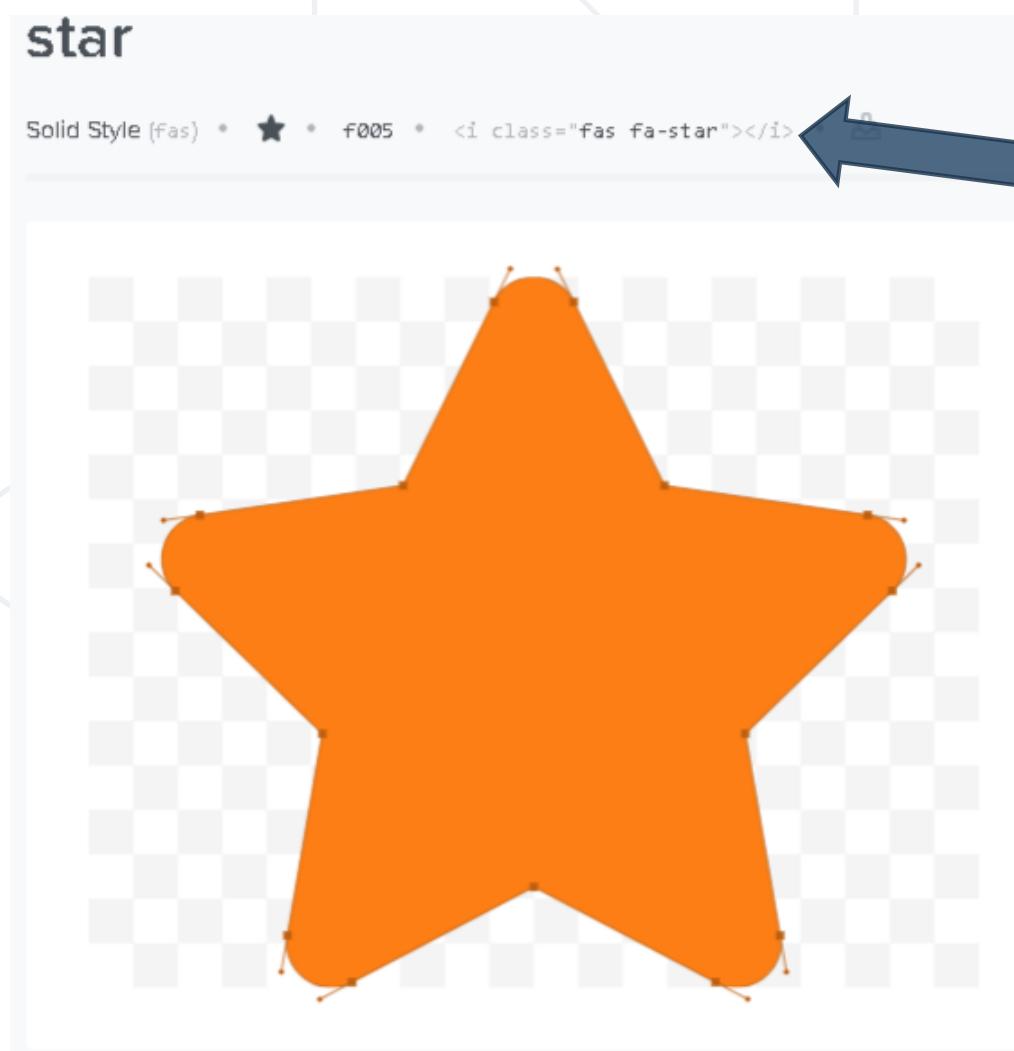
```
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.12.1/css/all.css">
```

- Import Font Awesome in the **css** file:

```
@import 'https://use.fontawesome.com/releases/v5.12.1/css/all.css';
```

- Font Awesome is designed to be used with inline elements. The **<i>** and **<span>** elements are widely used for icons

# Font Awesome Example



- Choose an icon
- Copy the *<i>* element
- Paste it in your HTML file

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="https://use.font
      awesome.com/releases/v5.12.1/css/all.css">
  </head>
  <body>
    <i class="fas fa-star"></i>
  </body>
</html>
```

- What is Typography?
- The principles of readability
- CSS properties: font-family, font-size, font-style, color, background
- Font Awesome Icons



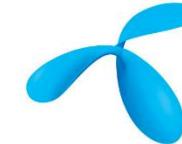
# SoftUni Diamond Partners



**XS**software



**SBTech**  
*we know sports*



telenor



**SoftwareGroup**  
*doing it right*

**NETPEAK**



**SmartIT**



**Postbank**

*Решения за твоето упре*

**SUPER  
HOSTING**  
**:BG**

**INDEAVR**  
*Serving the high achievers*

**INFRASTICS®**



**STEMO®**  
*Computer Systems & Software*



# SoftUni Organizational Partners



ИНФОРМАЦИОННО  
ОБСЛУЖВАНЕ

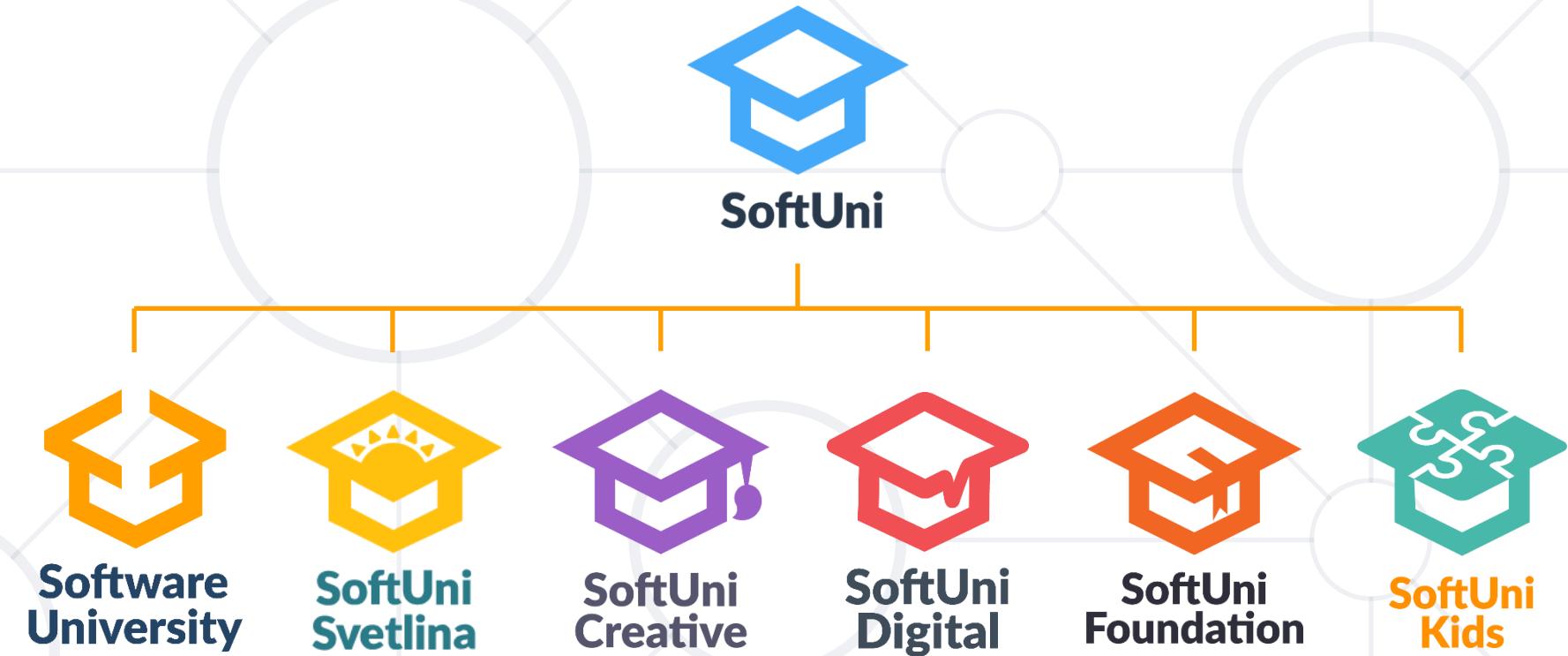
OneBit  
SOFTWARE



Lukanet.com



# Questions?



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>



# Trainings @ Software University (SoftUni)



- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Software University Forums
  - [forum.softuni.bg](http://forum.softuni.bg)



Software  
University



# CSS BOX MODEL



SoftUni Team  
Technical Trainers



SoftUni



Software University  
<https://softuni.bg>

# Table of Contents

1. CSS Box Model
2. Block and Inline Elements
3. Width and Height
4. Padding, Margin and Border
5. Box Sizing



Have a Question?



sli.do

#HTML-CSS

# CSS Box Model



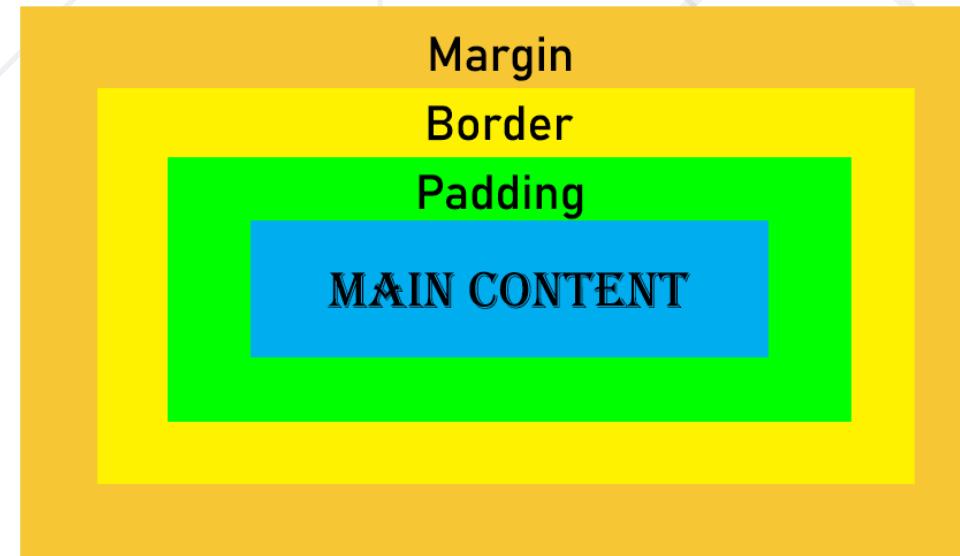
# What is CSS Box Model?

- The CSS box model is essentially a **box** that wraps around every HTML element
- All HTML elements can be considered as boxes
- The term "**box model**" is used when talking about design and layout
- CSS box model consists of: margins, borders, padding, and the actual content



# Parts of a box

- Content Box – the area where your content is displayed, which can be sized using properties like width and height
- Padding Box – the padding sits around the content
- Border Box – the border box wraps the content and any padding
- Margin Box – the margin wrapping the content, padding and border



# Block and Inline Elements



# Block and Inline Elements

- HTML is made up of various elements that act as the **building blocks** of web pages
- CSS has two different types of boxes — **block** and **inline**
  - Block Elements
  - Inline Elements
  - Inline-block Elements

# Block Elements

- Block element: starts on a **new line**, and fills up the horizontal space left and right on the web page
- Some examples of block elements are:
  - main, header, article, section, fieldset, nav, ul, ol, li, form, h1-h6, p, div
- You can add **margins** and **padding** on **all four sides** of any block element

# Block Elements - Example

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width" />
    <title></title>
  </head>
  <body>
    <div>This is my div tag.</div>
    <p>This is my paragraph tag.</p>
  </body>
</html>
```



# Inline Elements

- Inline element: **don't start** on a new line. They appear on the same line as the content and tags beside them
- Some examples of inline-block elements are:
  - a, label, map, span, strong, em, i, img, textarea, input, button, select
- You can add margins and padding just on **right** and **left** sides of any inline element

# Inline Elements - Example

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width" />
    <title></title>
  </head>
  <body>
    <div>This is my <strong>div</strong> tag.</div>
    <p>This is my <span>paragraph</span> tag.</p>
  </body>
</html>
```

This is my **div** tag.

This is my paragraph tag.

# Inline-Block Elements

- Inline-block elements are similar to inline elements
- They can have padding and margins added on **all four sides**
- You have to declare **display: inline-block** in your CSS code
- One common use for using inline-block is for creating navigation links horizontally

# Inline-Block Elements - Example

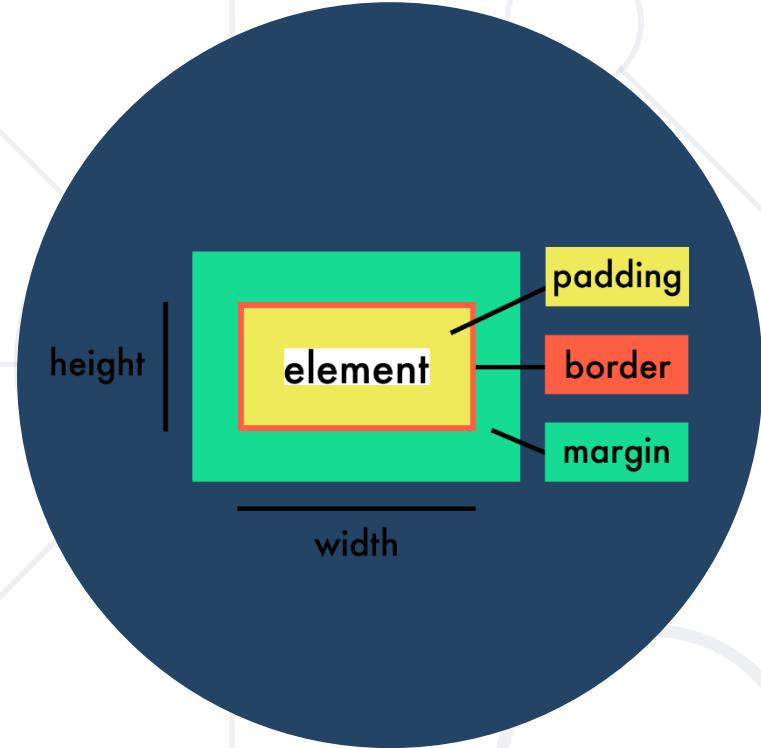
```
<ul>
  <li>Home</li>
  <li>About Us</li>
  <li>Clients</li>
  <li>Contacts</li>
</ul>
```

```
ul {
  background-color: #F0B27A;
  padding: 20px;
  list-style-type: none;
  text-align: center;
}

li {
  display: inline-block;
  padding: 0 20px;
  font-size: 20px;
}
```

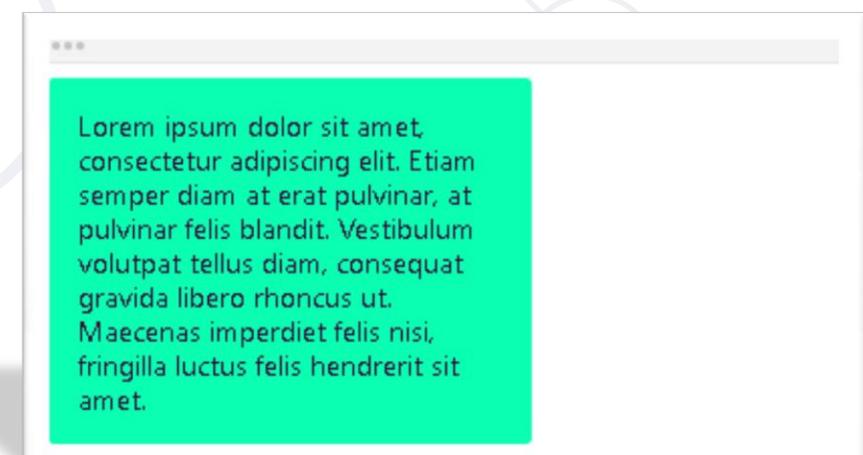
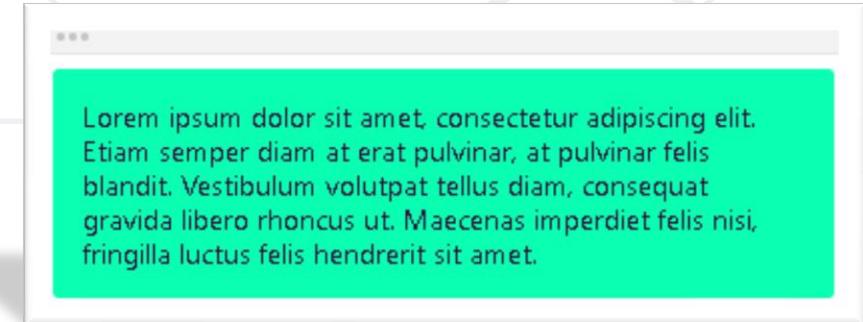
Home    About Us    Clients    Contact

# Width and Height



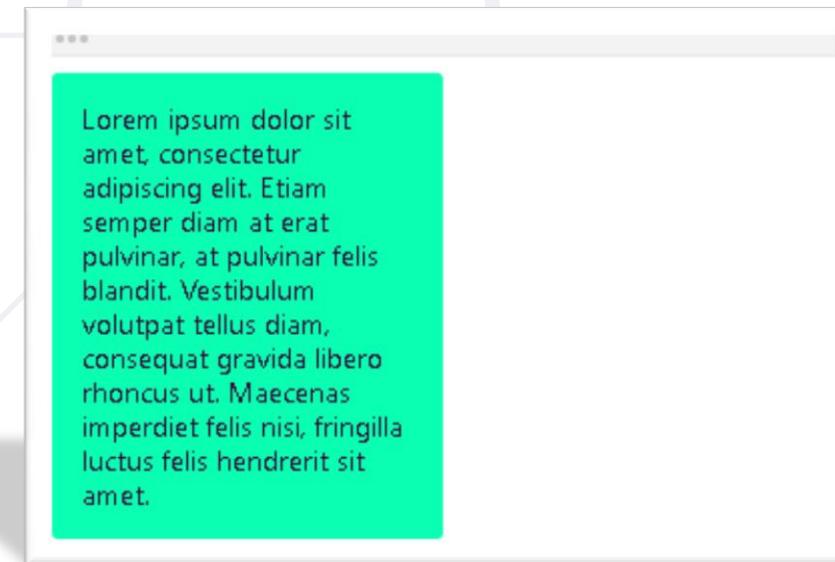
# Width

- Width – defines the width of the element
  - **width: auto;** - the element will **automatically** adjust its width to allow its content to be displayed correctly
  - **width: 240px;** - you can use numeric values like **pixels, (r)em, percentages...**



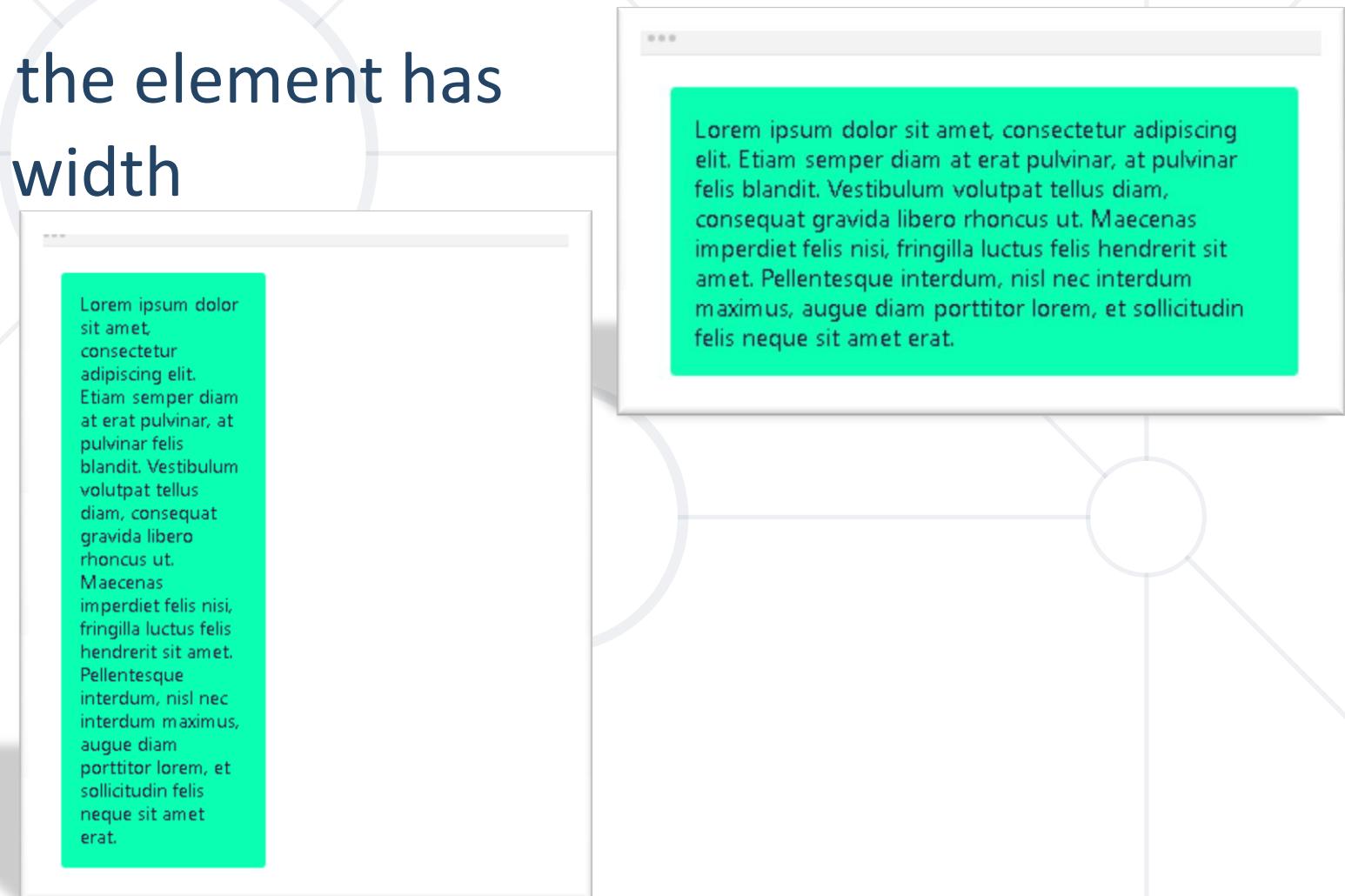
# Width

- **width: 50%;** - if you use **percentages**, the value is relative to the container's width

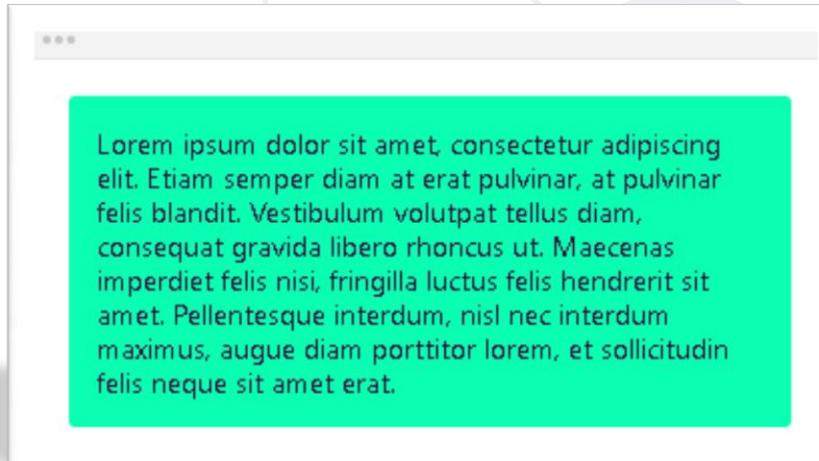


# Max-width

- Max-width – defines the **maximum** width the element can be
  - **max-width: none;** - the element has **no limit** in terms of width
  - **max-width: 150px;**



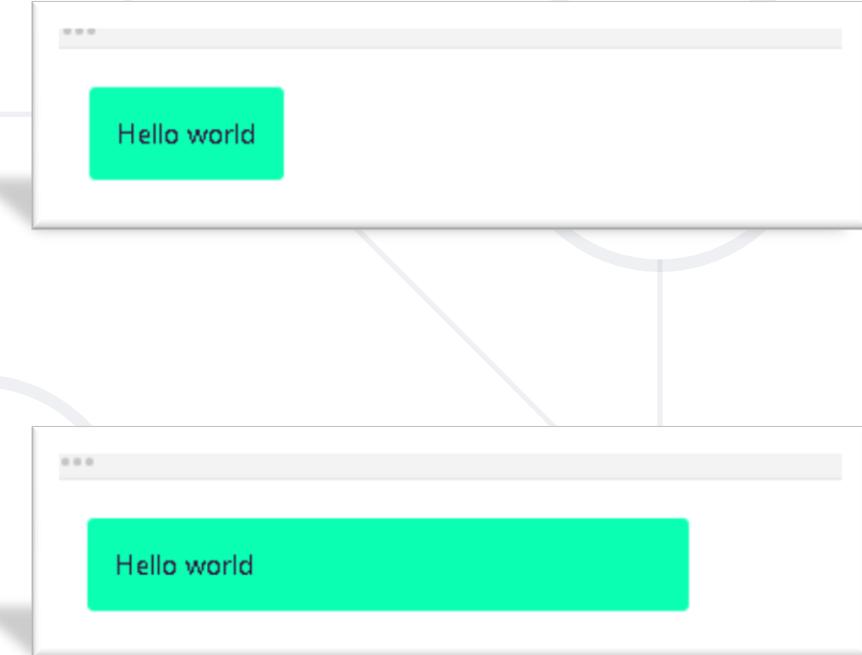
- **max-width: 2000px;** - you can use numeric values like **pixels**, **(r)em**, **percentages**...
- If the *maximum* width is **larger** than the element's *actual* width, the max width has **no effect**



...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

# Min-width

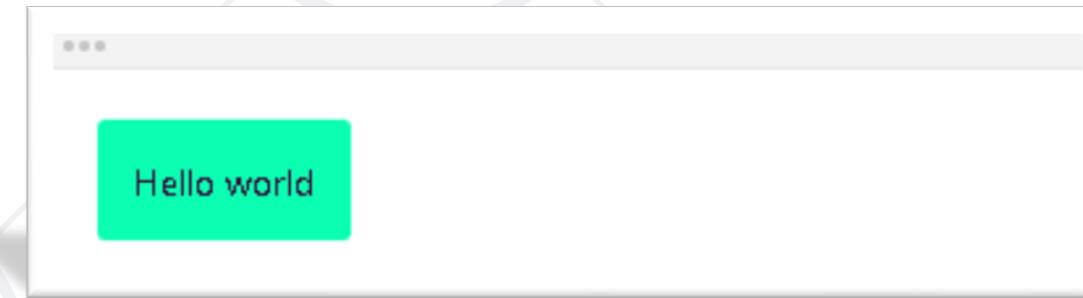
- Min-width – defines the **minimum** width the element
  - **min-width: 0;** - the element has **no minimum** width
  - **min-width: 300px;** - if the *minimum* width is **larger** than the element's *actual* width, the min width will be applied



# Min-width

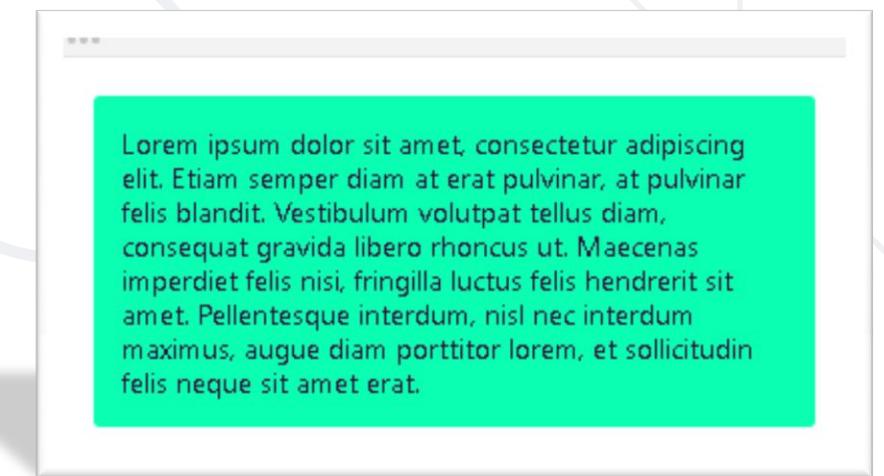
- **min-width: 5px;**

If the *minimum* width is **smaller** than the element's *actual* width,  
the min width has **no effect**



# Height

- Height – defines the height of the element
  - **height: auto;** - the element will **automatically** adjust its height to allow its content to be displayed correctly

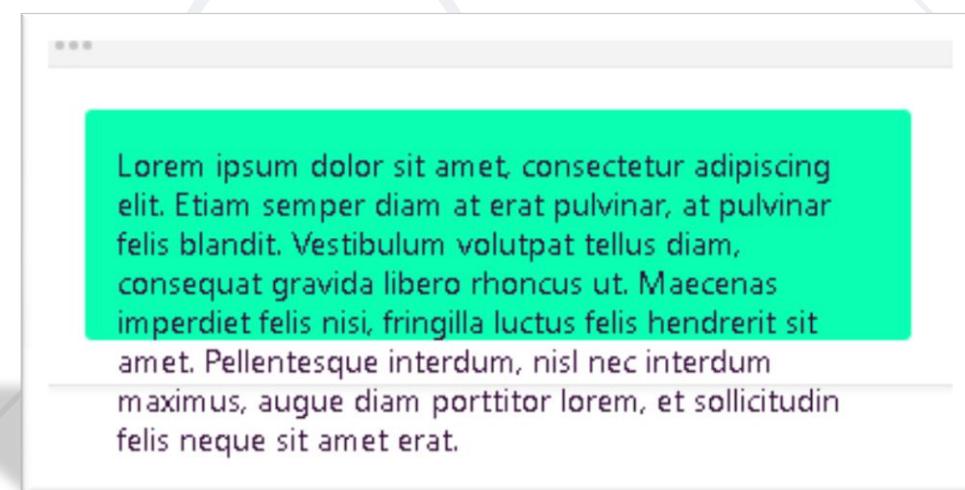


Placeholder text for the green box:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

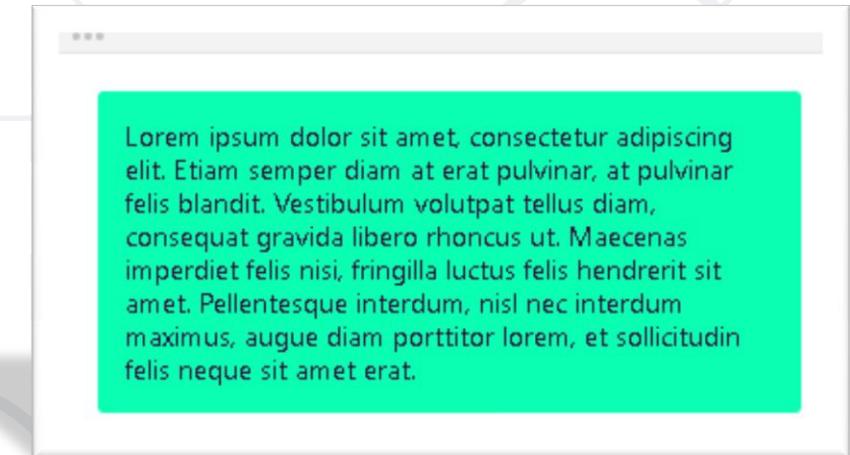
# Height

- **height: 100px;** - you can use numeric values like **pixels, (r)em, percentages...**
- If the content does not fit within the specified height, it will **overflow**
- How the container will handle this overflowing content is defined by the **overflow property**



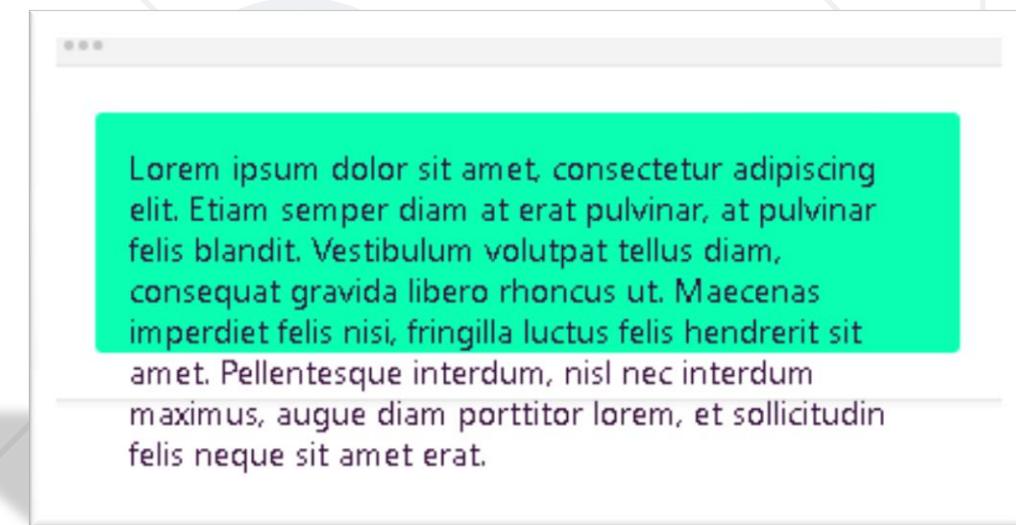
# Max-height

- Max-height – defines the maximum height the element can be
  - **max-height: none;** - the element has **no limit** in terms of height
  - **max-height: 2000px;** - if the *maximum* height is **larger** than the element's *actual* height, the max height has **no effect**



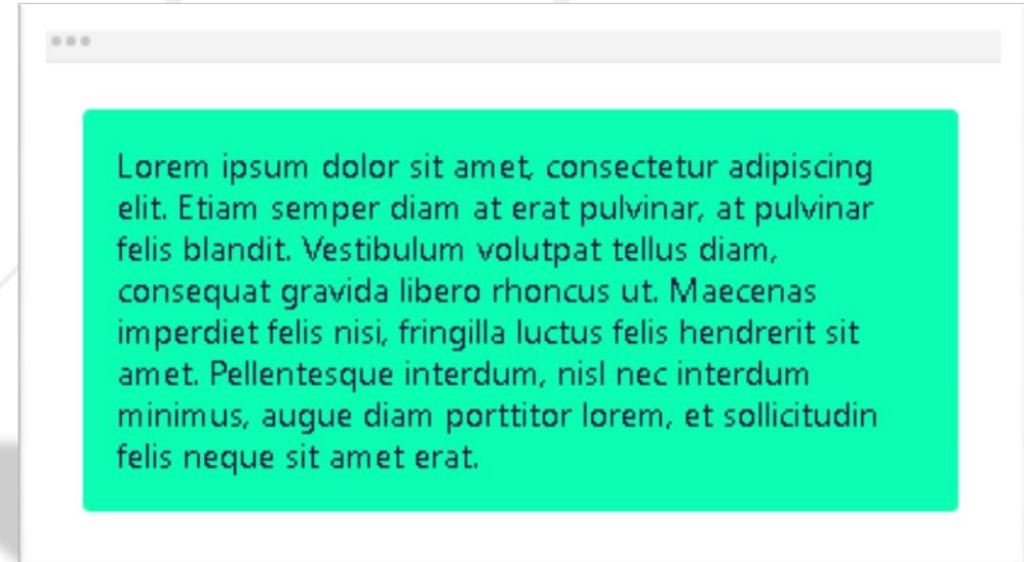
# Max-height

- **max-height: 100px;**
- If the content does not fit within the maximum height, it will **overflow**
- How the container will handle this overflowing content is defined by the **overflow property**



# Min-height

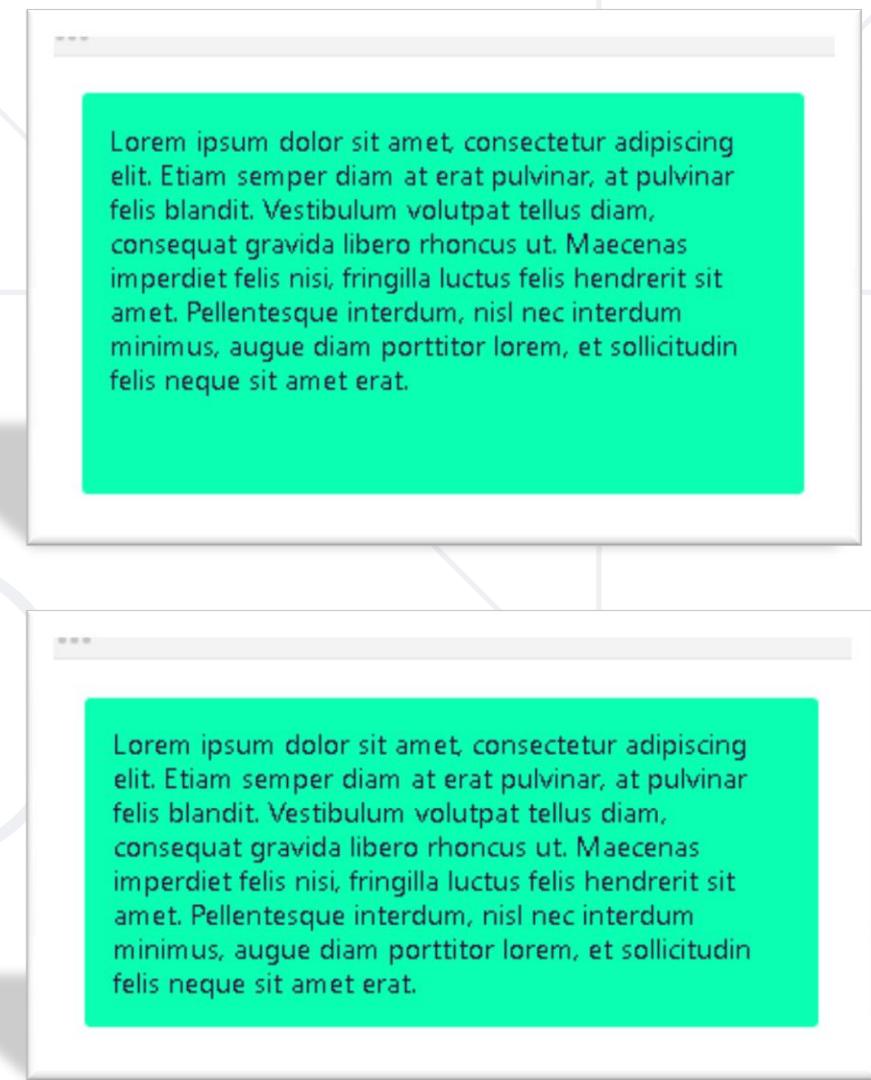
- Min-height – defines the minimum height the element
  - **min-height: 0;** - the element has **no minimum** height



...  
  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum minimus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

# Min-height

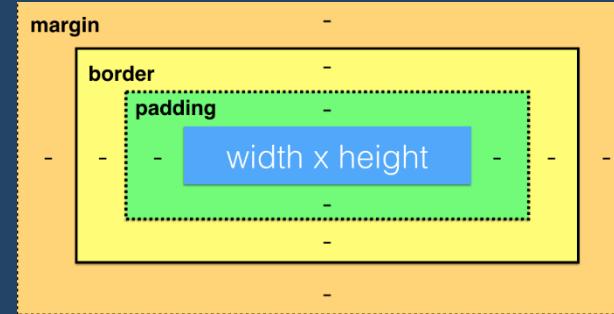
- **min-height: 200px;** - if the *minimum height* is **larger** than the element's *actual height*, the min height will be applied
- **min-height: 5px;** - if the *minimum height* is **smaller** than the element's *actual height*, the min height has **no effect**



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum minimus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut. Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum minimus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

# Margin, Padding and Border



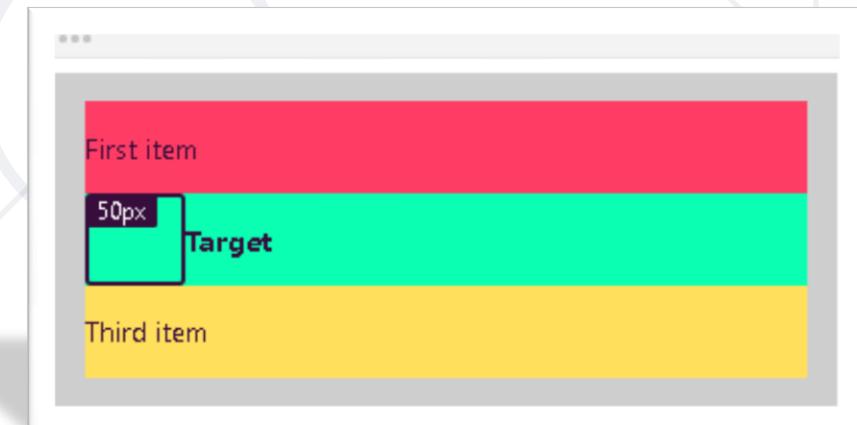
# Padding

- **Padding** – defines the space **inside** the element
- Padding-bottom:
  - padding-bottom: 0;
  - padding-bottom: 50px;



# Padding

- Padding-left:
  - padding-left: 0;
  - padding-left: 50px;



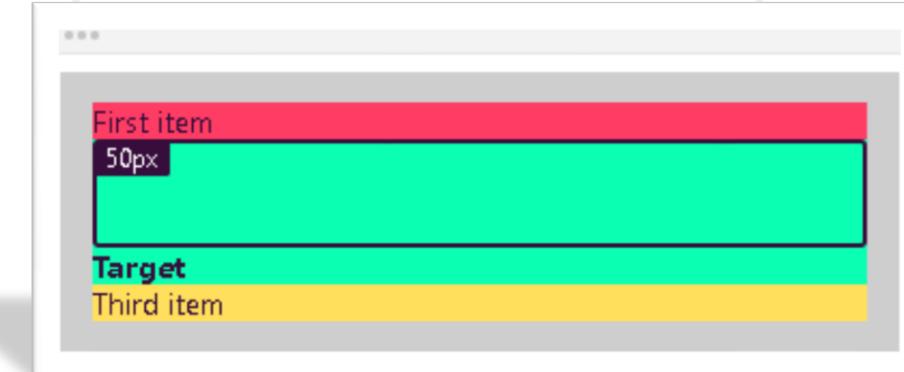
# Padding

- Padding-right:
  - padding-right: 0;
  - padding-right: 50px;



# Padding

- Padding-top:
  - padding-top: 0;
  - padding-top: 50px;

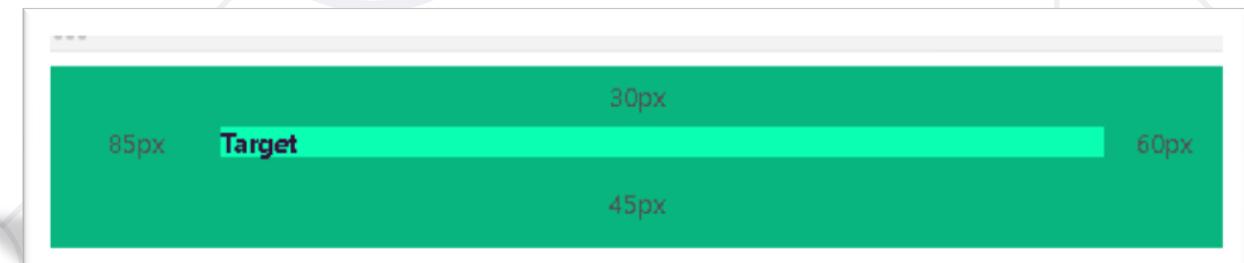
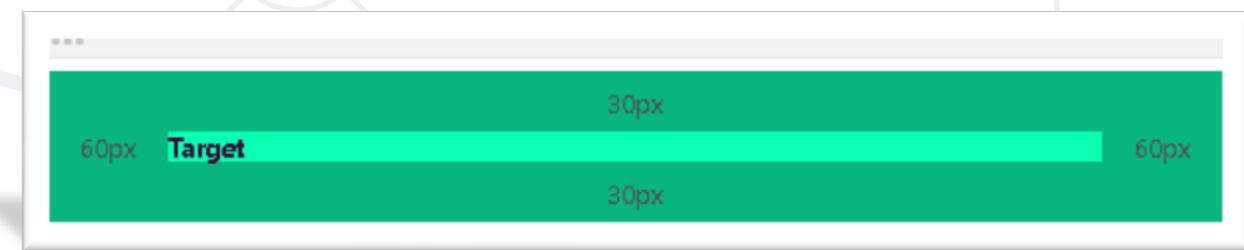


- Padding: shorthand property for padding-top, padding-right, padding-bottom, padding-left
  - `padding: 0;`
  - `padding: 30px;`

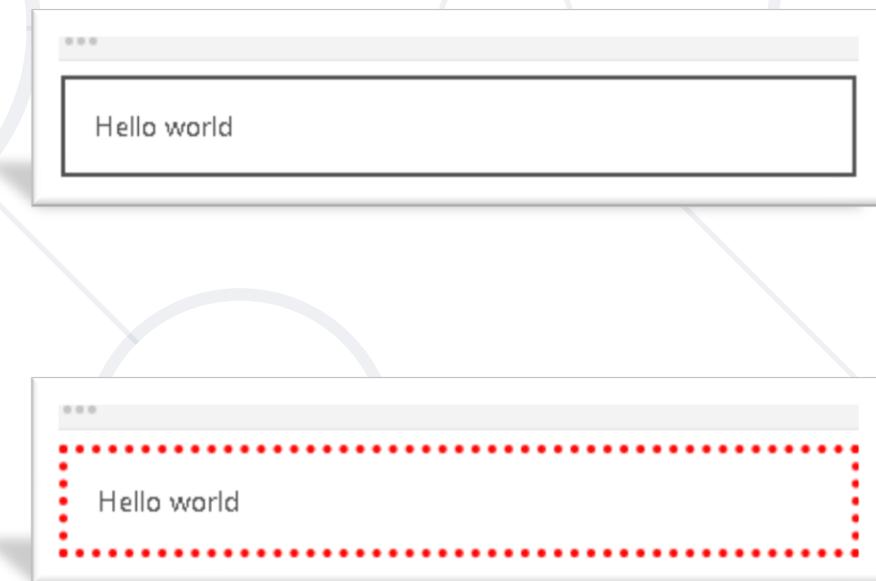


# Padding

- padding: 30px 60px;
- padding: 30px 60px 45px;
- padding: 30px 60px 45px 85px;



- Border: shorthand property for **border-width**, **border-style** and **border-color**
  - border: 2px solid black;
  - border: 4px dotted red;



# Margin

- **Margin** – defines the space **outside** the element
- Margin-bottom:
  - margin-bottom: 0;
  - margin-bottom: 30px;



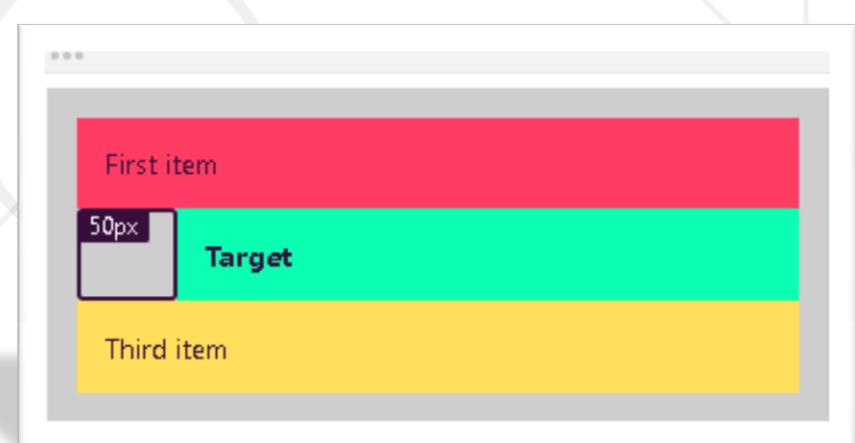
# Margin

- Margin-left:

- margin-left: 0;



- margin-left: 50px;



# Margin

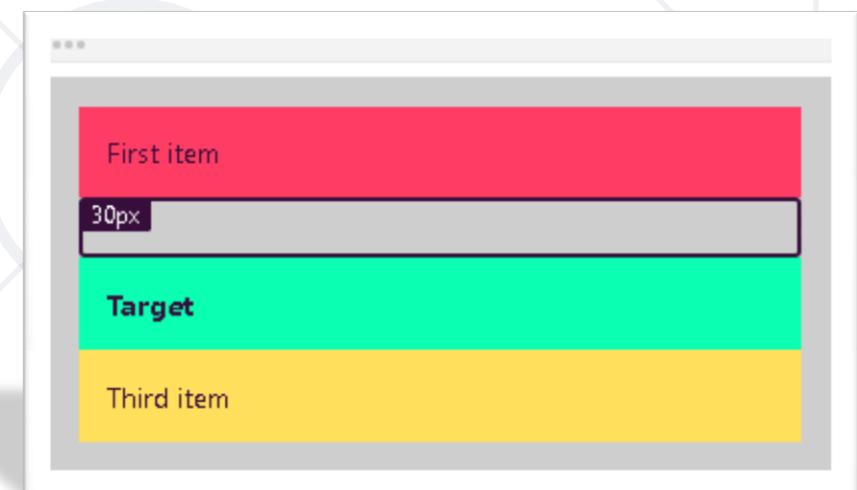
- Margin-right:
  - margin-right: 0;
  - margin-right: 50px;



# Margin

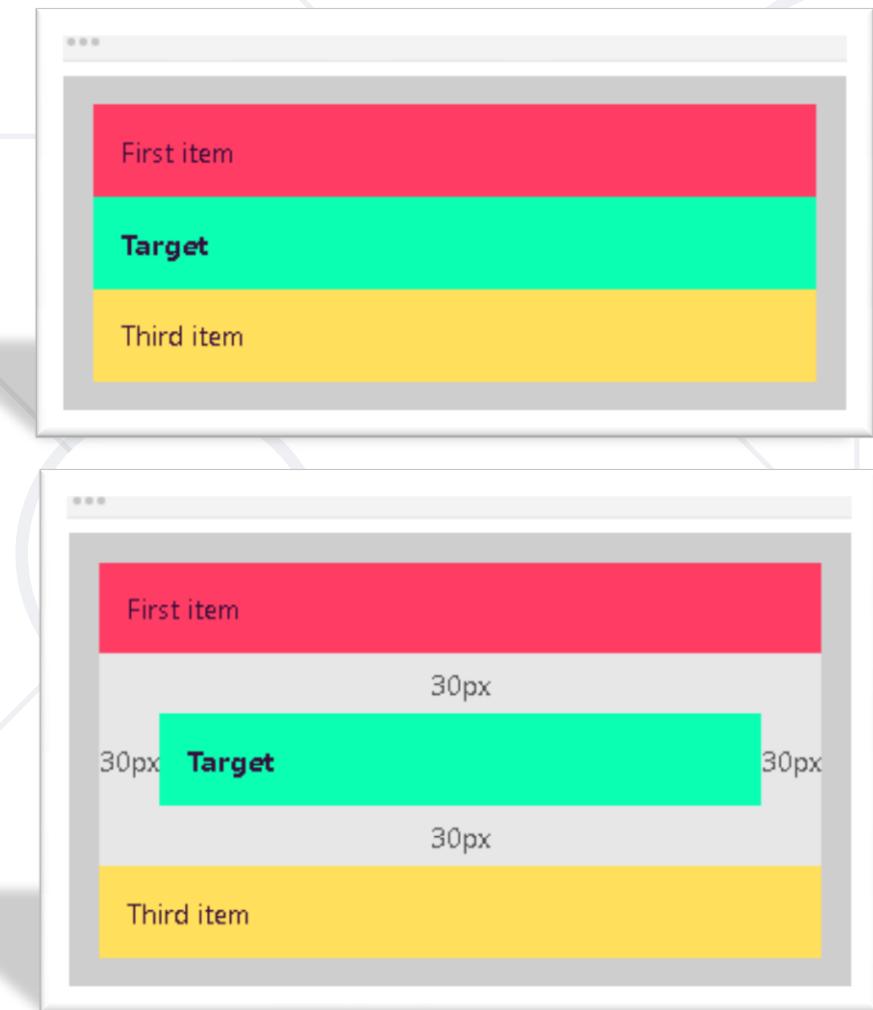
- Margin-top:

- margin-top: 0;



# Margin

- Margin: shorthand property for margin-top, margin-right, margin-bottom, margin-left
  - margin: 0;
  - margin: 30px;



- **Cursor** – Sets the mouse cursor when hovering the element

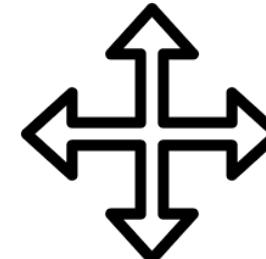
- Pointer

```
p {  
  cursor: pointer  
}
```

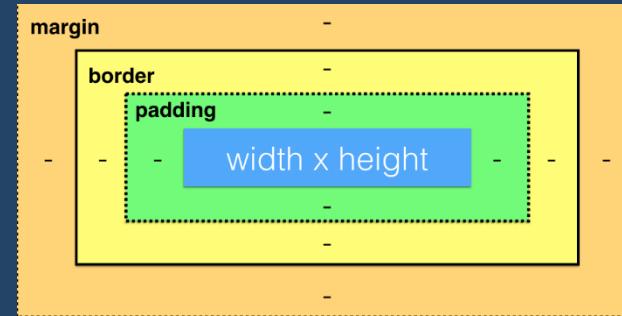


- Move

```
p {  
  cursor: move;  
}
```



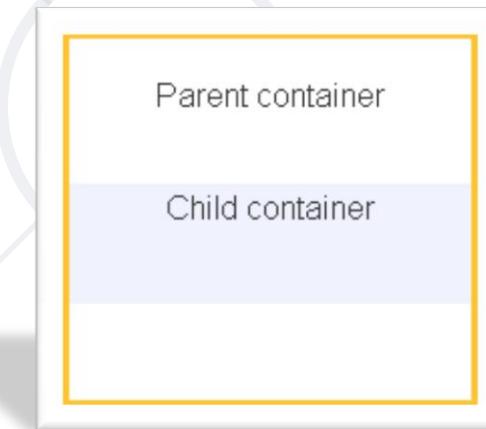
# Box Sizing



# Box-sizing

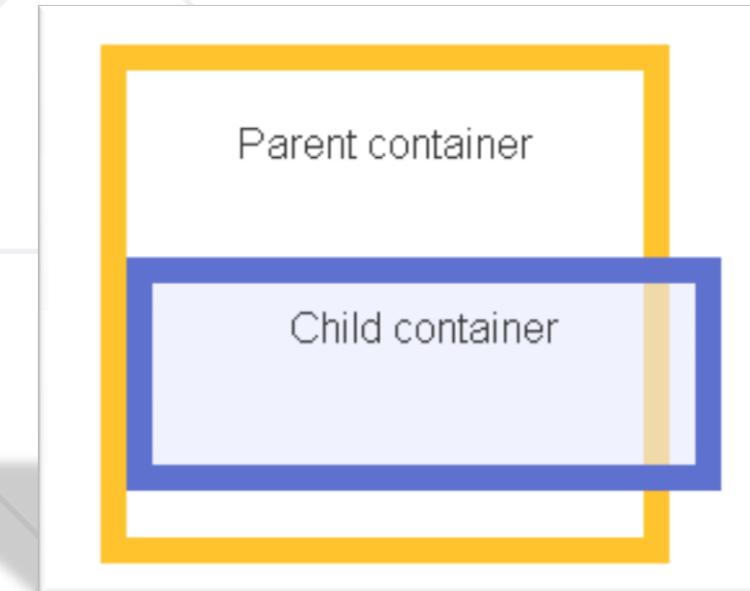
- Box-sizing: sets how the total width and height of an element is calculated
  - **content-box** – initial and default value
  - The **width** and **height** properties include the content, but does **not include** the padding, border and margin

```
box-sizing: content-box;  
width: 200px;
```



# Box-sizing

```
box-sizing: content-box;  
width: 200px;  
border: 10px solid #5B6DCD;  
padding: 5px;
```

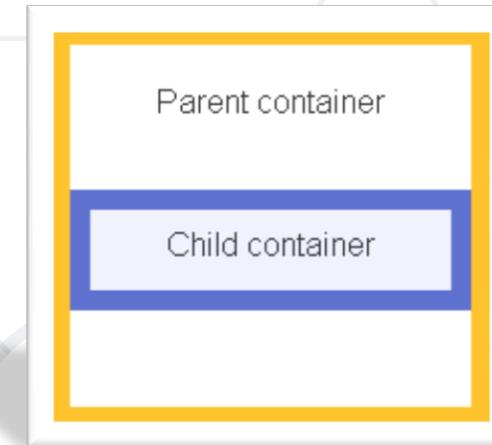


- The full width is:  $200\text{px} + 2*10\text{px} + 2*5\text{px} = 230\text{px}$

# Box-sizing

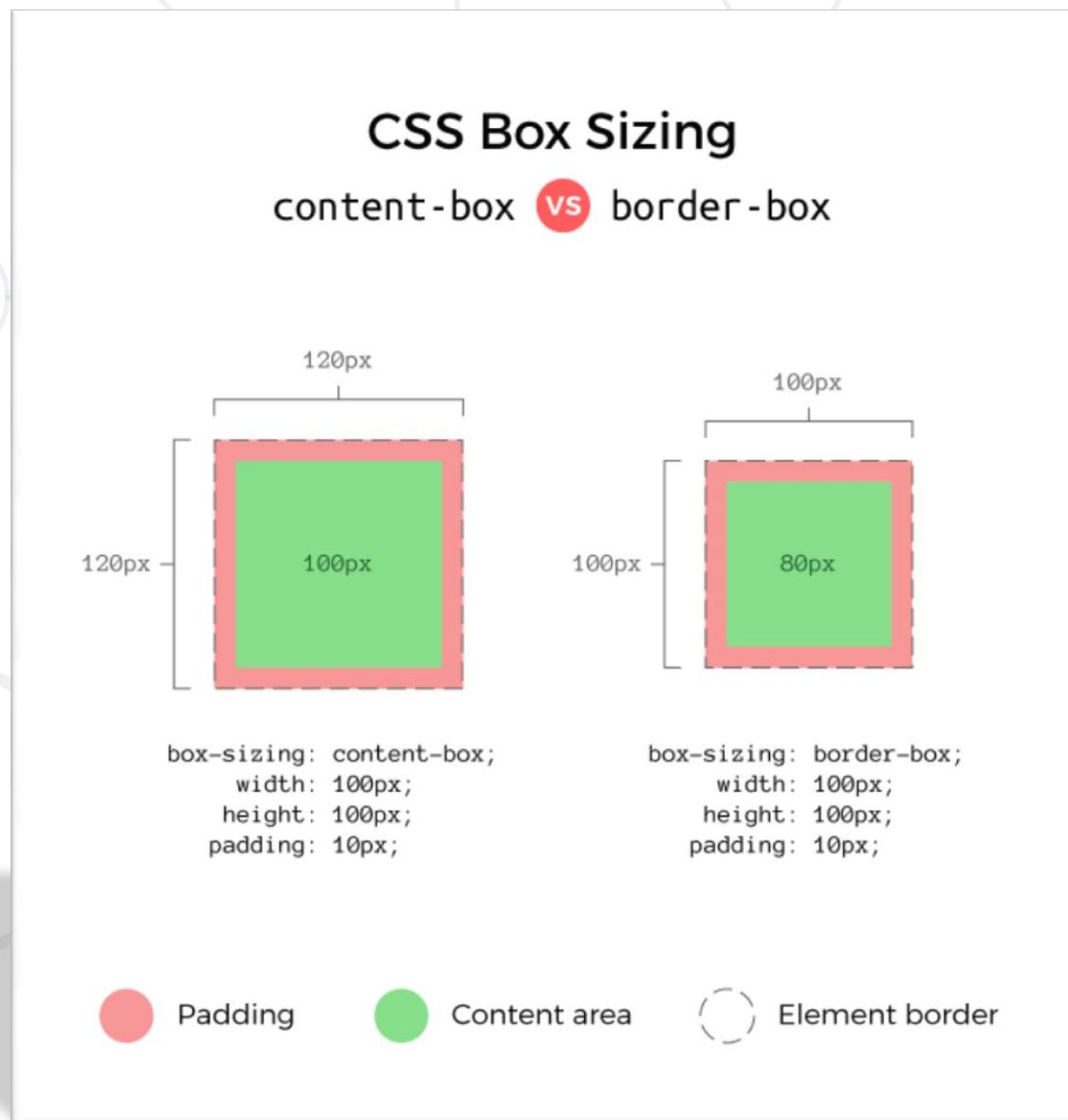
- **border-box** – the **width** and **height** of the element apply to all parts of the element: the **content**, the **padding** and the **borders**

```
box-sizing: border-box;  
width: 200px;  
border: 10px solid #5B6DCD;  
padding: 5px;
```



- The full width is **200px**
- The content width is equal to:  $200px - 2*10px - 2*5px = 170px$

# Content-box vs Border-box



# Universal Box-sizing

- The box-sizing **Reset** takes care of the box-sizing of every element by setting it to border-box using universal CSS selector
- Save your **time** and don't write the same thing **again-and-again**
- Set the "universal box-sizing" with inheritance:

```
html {  
  box-sizing: border-box;  
}  
  
*, *:before, *:after {  
  box-sizing: inherit;  
}
```

- **What is Box Model?**
- **Width and Height to the elements**
- **What are the padding, border and margin?**
- **What is box-sizing?**
- **How to reset Box-sizing?**



# SoftUni Diamond Partners



**xssoftware**



**SBTech**  
*we know sports*



**telenor**



**SoftwareGroup**  
*doing it right*

**NETPEAK**



**SmartIT**



**Postbank**

*Решения за твоето упре*

**SUPER  
HOSTING**  
**:BG**

**INDEAVR**  
*Serving the high achievers*

**INFRASTICS®**



**STEMO®**  
*Computer Systems & Software*



# SoftUni Organizational Partners



ИНФОРМАЦИОННО  
ОБСЛУЖВАНЕ

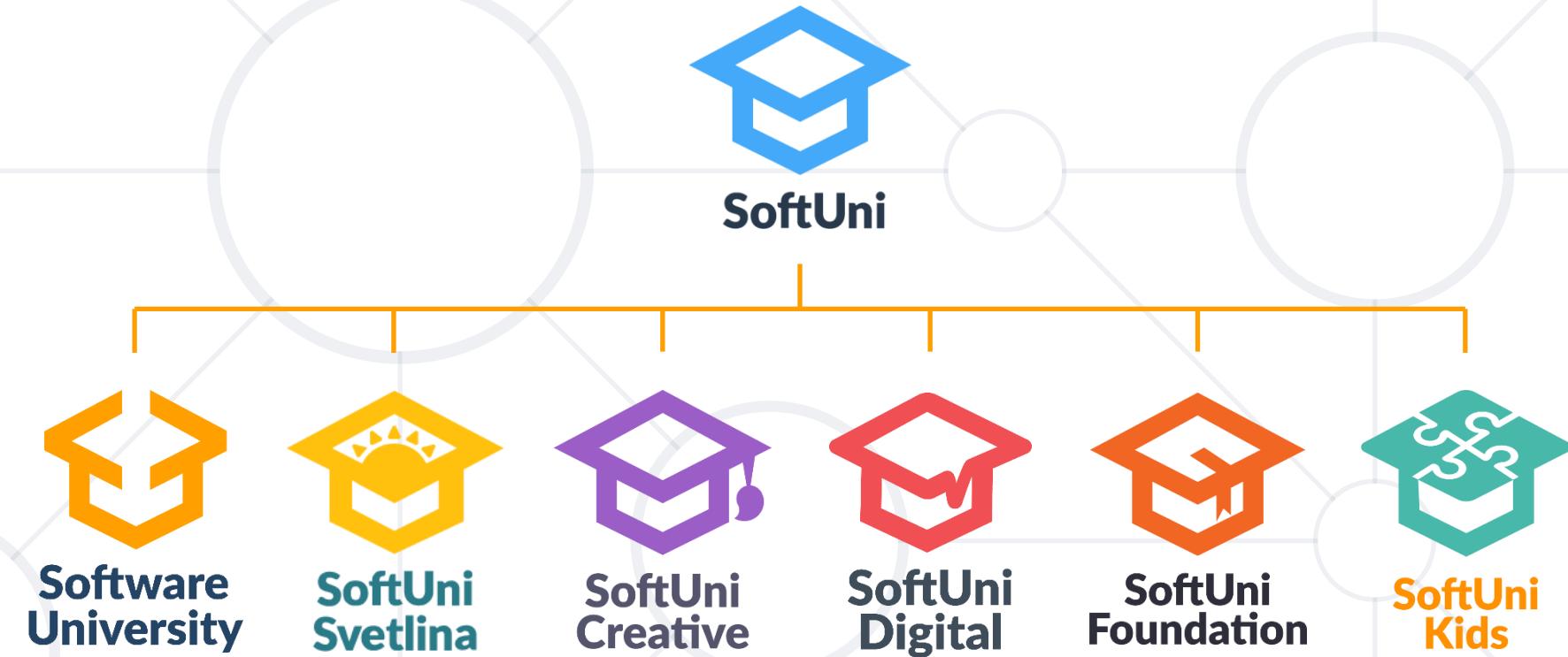
OneBit  
SOFTWARE



Lukanet.com



# Questions?



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>



# Trainings @ Software University (SoftUni)



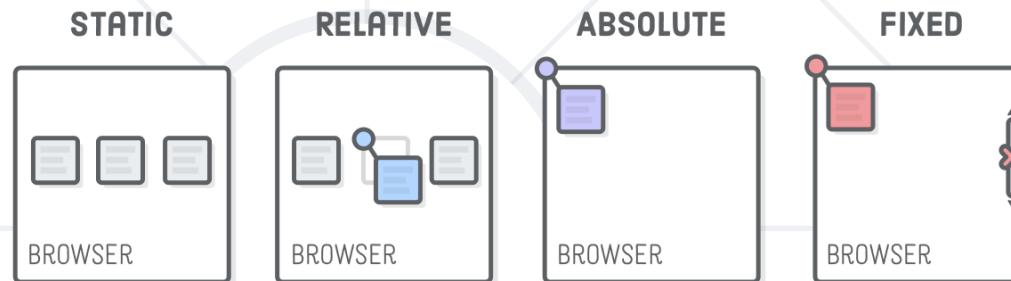
- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Software University Forums
  - [forum.softuni.bg](http://forum.softuni.bg)



Software  
University



# POSITION & FLOAT



SoftUni Team  
Technical Trainers



# SoftUni

Software University  
<https://softuni.bg>



# Table of Contents

1. Float property
2. Clear property
3. Position: static, relative, absolute, fixed and sticky
4. Positioning Properties
5. Z-index

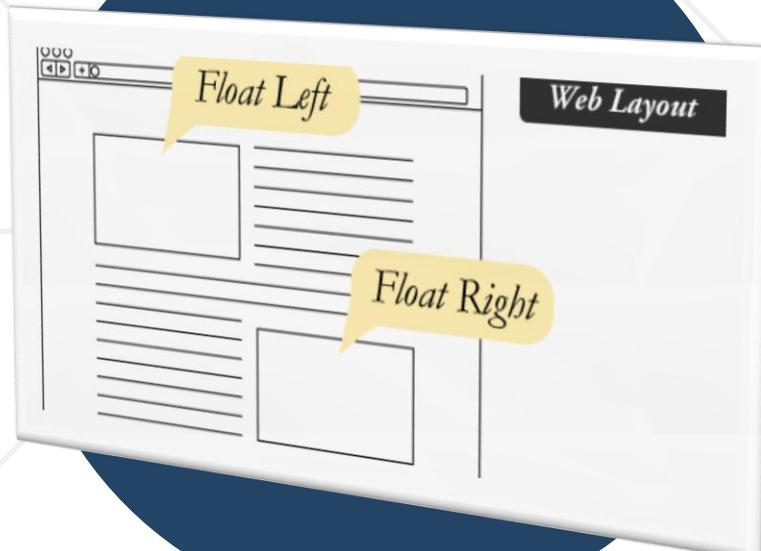


Have a Question?

**sli.do**

**#HTML-CSS**

# Float Property



# What is Float?

- CSS float is a property that forces any element to float (right, left, none, inherit) inside its parent body with the rest of the element to wrap around it
- The element is removed from the normal flow of the page, though still remaining a part of the flow

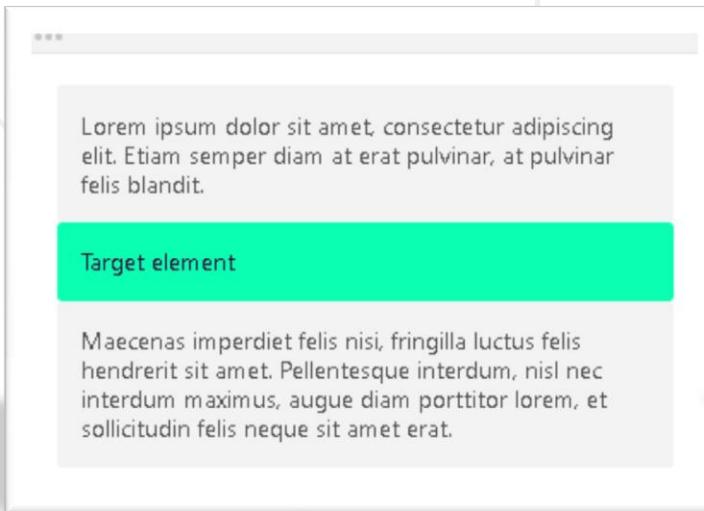


# Float Property

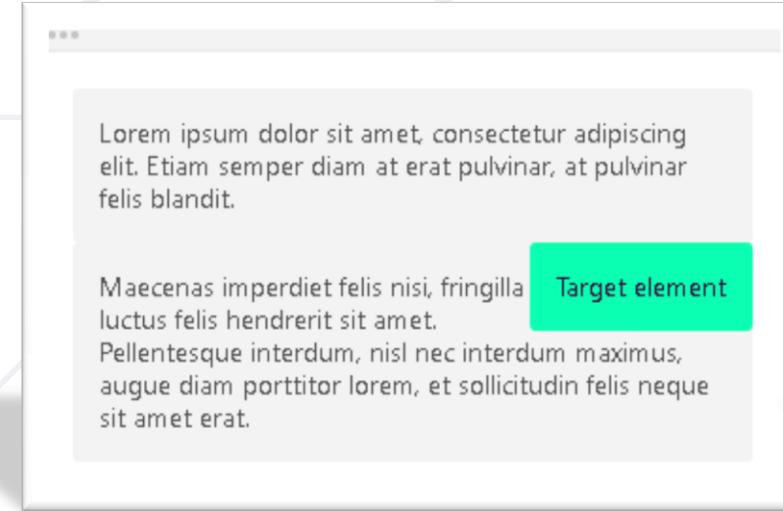
- Float values:
  - `float : right;` - floats the element to right of it's container
  - `float : left;` - floats the element to left of it's container
  - `float : none;` - it will restrict the element to float
  - `float : initial;` - the element remains to it's default position
  - `float : inherit;` - enables the element to inherit the property from its parent element

# Float Property - Example

- float: none;



- float: right;



- float: left;



# Floating multiple Elements

```
<body>
  
  
  
  
</body>
```

```
.image {
  float: left;
  width: 150px;
  height: 100px;
  margin: 5px;
}
```



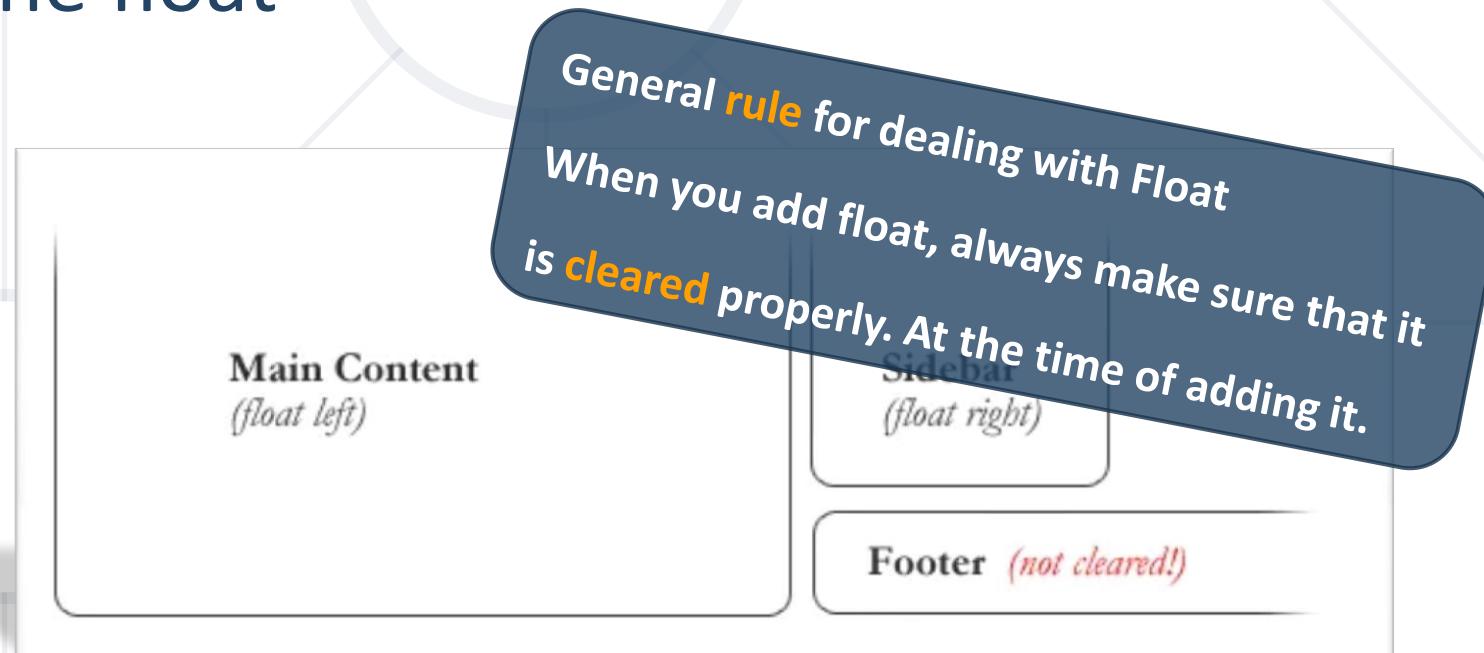
# Float Property

- As float implies the use of the block layout, it modifies the computed value of the display values, in some cases:

Specified value → Computed value  
inline → block  
inline-block → block

# Clearing the Float

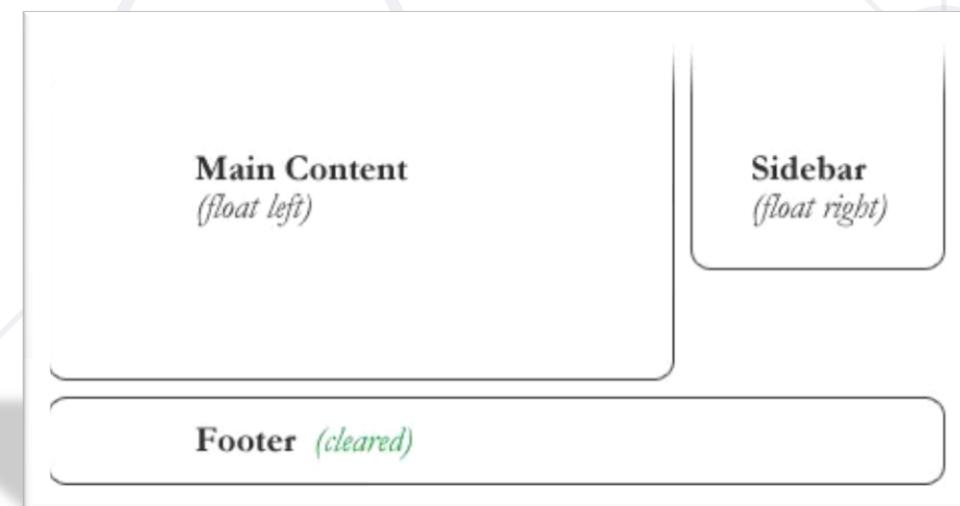
- Float's sister property is **clear**
- An element that has the clear property set on it will not move up adjacent to the float like the float desires but will move itself down past the float



# Clearing the Float

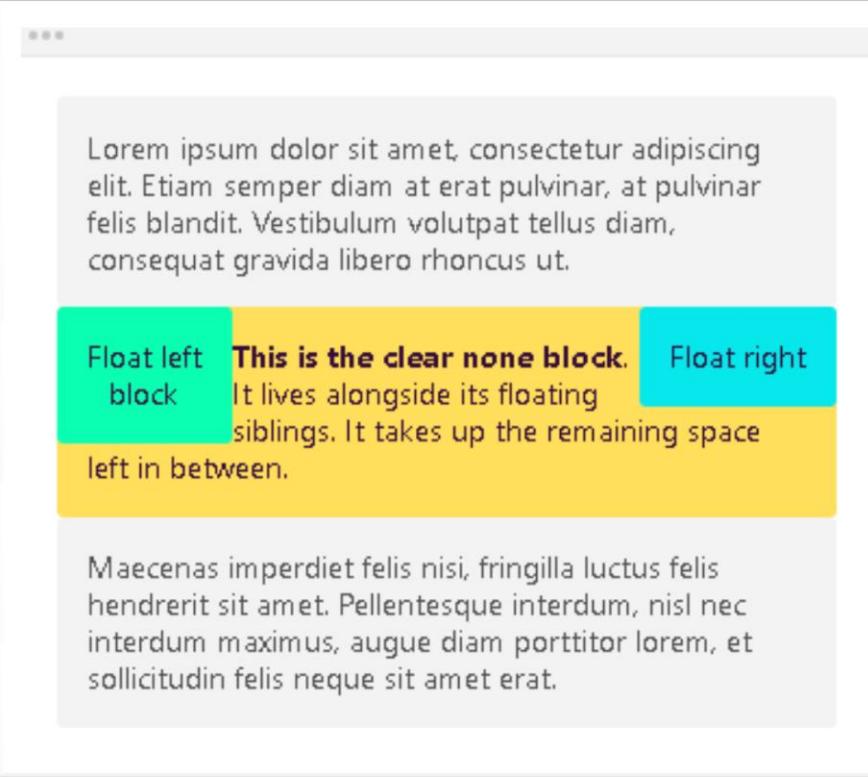
- In the above example, the **sidebar** is **floated** to the **right** and is shorter than the main content area
- The footer then is required to jump up into that available space as is required by the float
- To fix this problem, the **footer** can be **cleared** to ensure it stays beneath both floated columns

```
#footer {  
    clear: both;  
}
```



- The clear property has four values:

- clear: none;



...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

Float left block **This is the clear none block.** It lives alongside its floating siblings. It takes up the remaining space left in between.

Float right

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

- clear: left;



...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

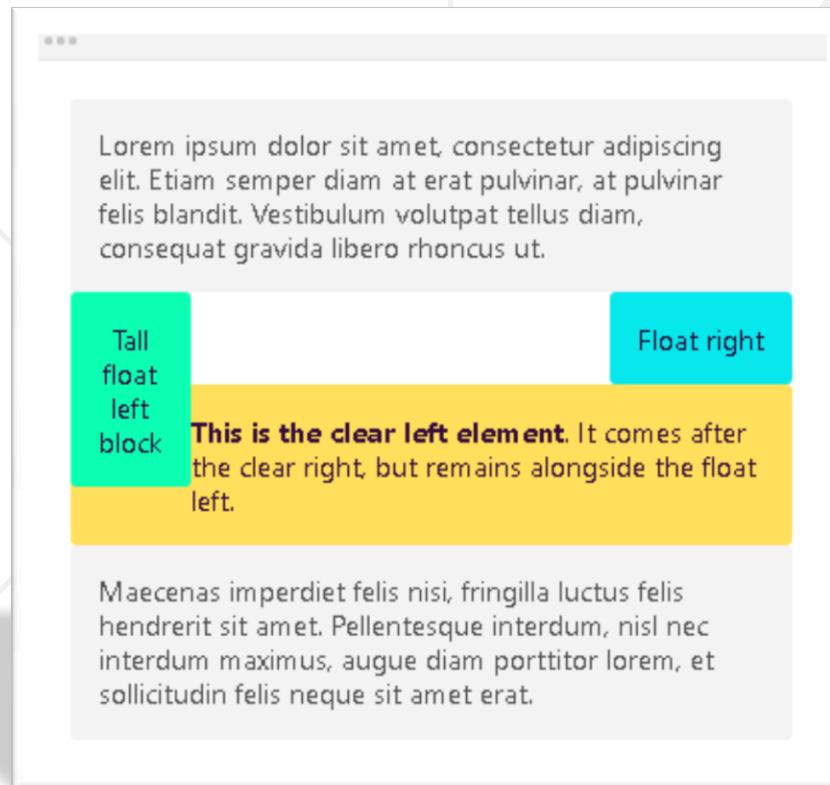
Float left

**This is the clear left element.** It comes after the clear left, but remains alongside the float right.

Tall float right block

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

- **clear: right;**



The diagram illustrates the effect of the CSS property `clear: right;`. It shows two columns of content. The left column contains a tall float-left block, followed by a yellow box labeled "This is the clear left element." The right column contains a float-right block. A callout points from the text in the yellow box to the right column, indicating that the clear left element remains alongside the float left.

...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

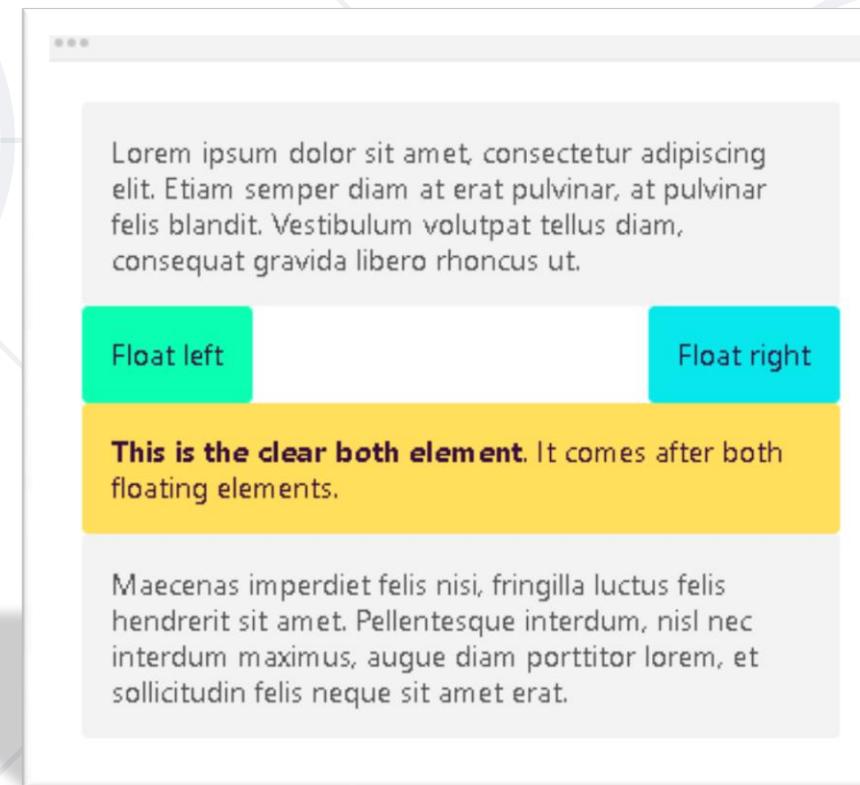
Tall float left block

Float right

This is the **clear left element**. It comes after the clear right, but remains alongside the float left.

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

- **clear: both;**



The diagram illustrates the effect of the CSS property `clear: both;`. It shows two columns of content. The left column contains a float-left block, followed by a yellow box labeled "This is the clear both element." The right column contains a float-right block. A callout points from the text in the yellow box to the right column, indicating that the clear both element follows both floating elements.

...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

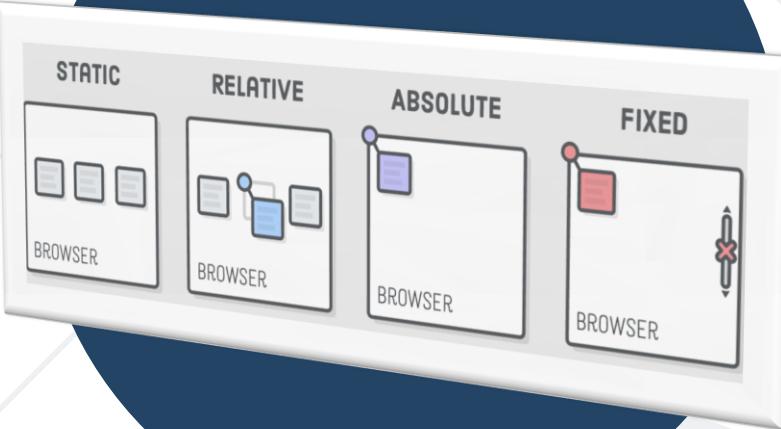
Float left

Float right

This is the **clear both element**. It comes after both floating elements.

Maecenas imperdiet felis nisi, fringilla luctus felis hendrerit sit amet. Pellentesque interdum, nisl nec interdum maximus, augue diam porttitor lorem, et sollicitudin felis neque sit amet erat.

# Position



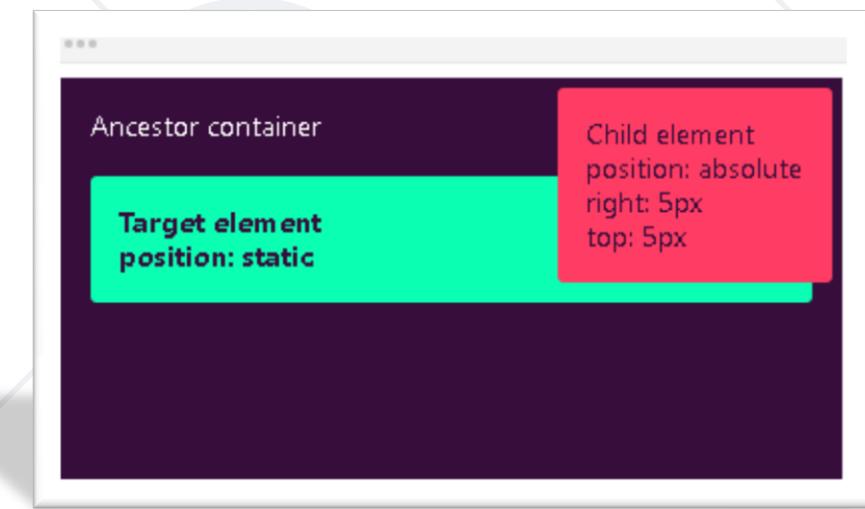
# Position

- The **position** property specifies the type of positioning method used for an element
  - Static
  - Relative
  - Absolute
  - Fixed
  - Sticky

# Position Static

- Static – the **default** state of every element. It just means "put the element into its **normal position** in the document layout flow"
- Also, it will **not** react to the following properties: top, bottom, left, right, z-index

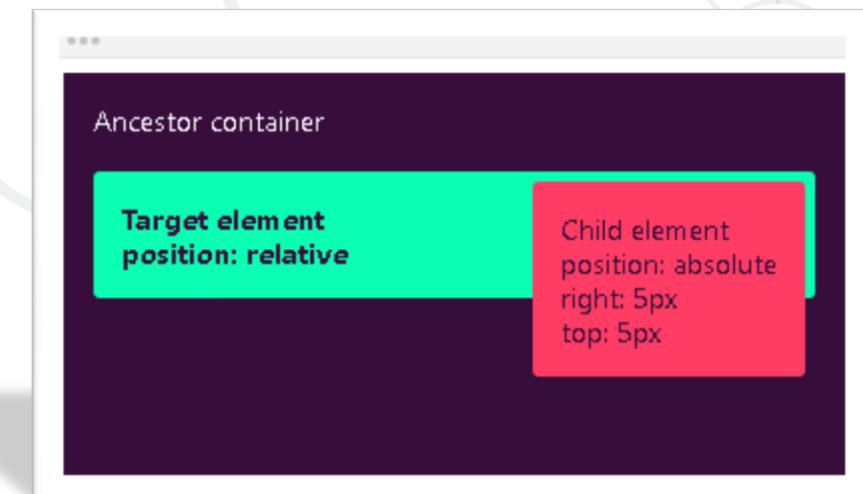
```
[selector] {  
    position: static;  
}
```



# Position Relative

- Relative – very similar to static positioning, **except** that once the positioned element has taken its place, you can then modify its final position with the **positional properties** – left, right, top, bottom, z-index
- It also makes the element **positioned**: it will act as **anchor** point for the absolutely positioned block

```
[selector] {  
    position: relative;  
}
```



# Position Absolute

- Absolute – the element will **not remain** in the natural flow of the page. It will **react** to the positional properties
- It will position itself according to the closest positioned ancestor
- Because it's positioned, it will act as an anchor point for the absolutely positioned block

```
[selector] {  
    position: absolute;  
}
```



# Position Fixed

- Fixed – the element will **not remain** in the natural flow of the page. It will **react** to the positional properties
- It will position itself according to the **viewport**
- Because it's positioned, it will act as an **anchor point** for the absolutely positioned block

```
[selector] {  
    position: fixed;  
}
```



# Position Sticky

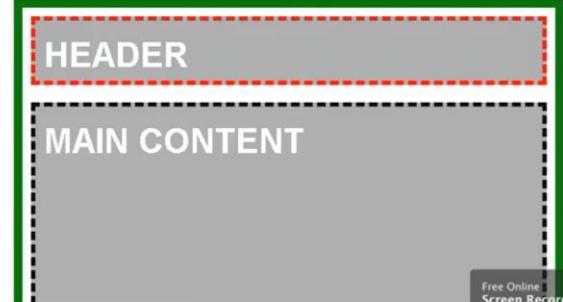
- Sticky – the element is positioned based on the user's **scroll position**
- A sticky element toggles between **relative** and **fixed**, depending on the scroll position
- It is positioned relative until a given offset position is met in the viewport – then it "sticks" in place (like position: fixed)

# Position Sticky - Example

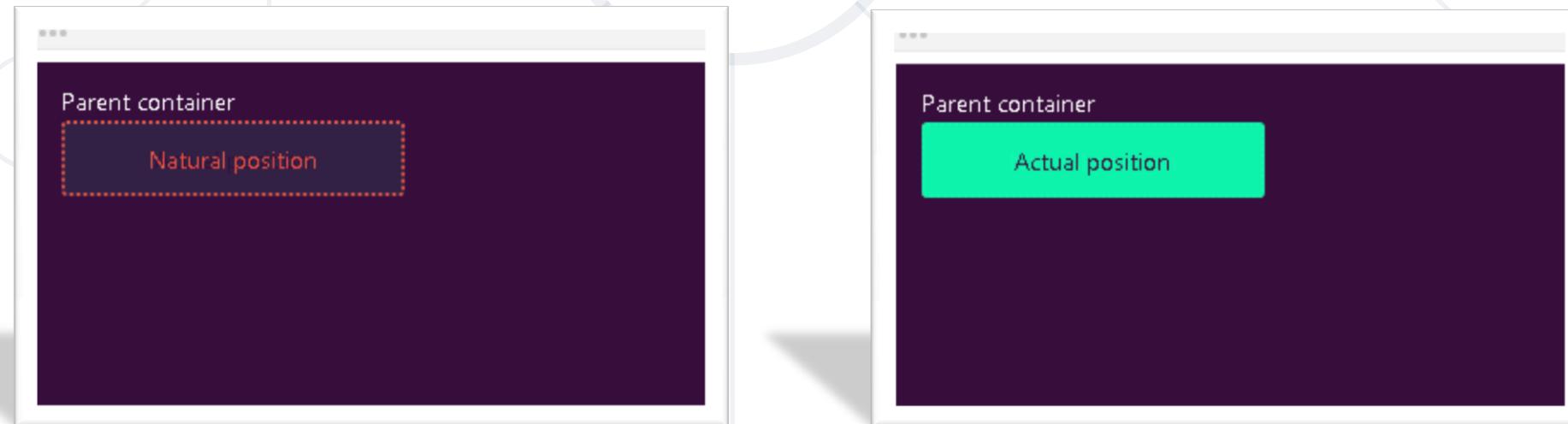
```
.main-container {  
    max-width: 600px;  
    margin: 0 auto;  
    border: 10px solid green;  
    padding: 10px;  
    margin-top: 40px;  
}  
  
.main-header,  
.main-content,  
.main-footer {  
    padding: 10px;  
    background: #aaa;  
    border: 5px dashed #000;  
    margin: 20px 0;  
}
```

```
<main class="main-container">  
    <header class="main-header">HEADER</header>  
    <div class="main-content">MAIN CONTENT</div>  
    <footer class="main-footer">FOOTER</footer>  
</main>
```

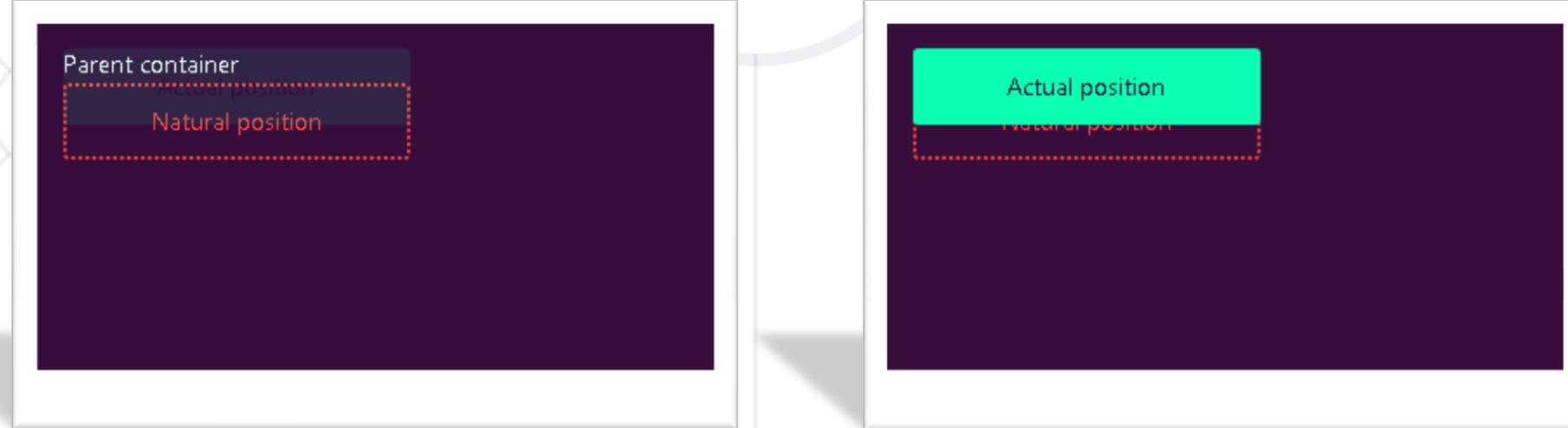
```
.main-content{  
    min-height: 1000px;  
}  
  
.main-header{  
    height: 50px;  
    border-color: red;  
    position: sticky;  
    top: 0;  
}
```



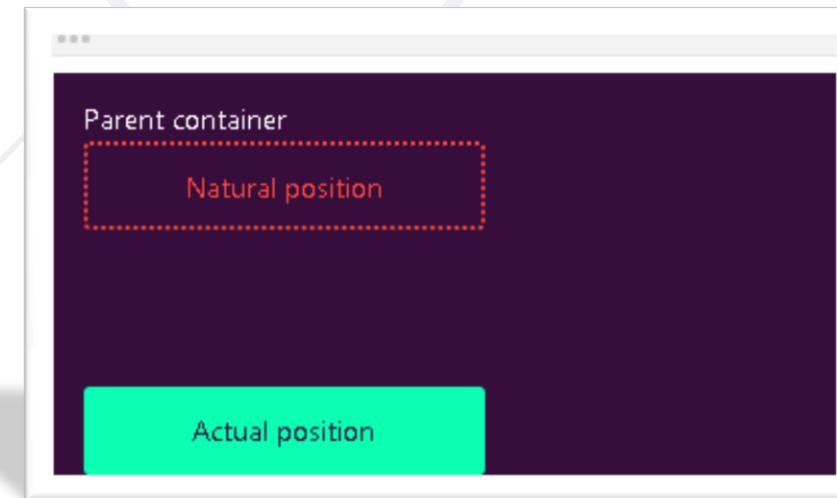
- Bottom – defines the position of the element according to its **bottom** edge
  - `bottom: auto;` - the element will remain in its **natural** position



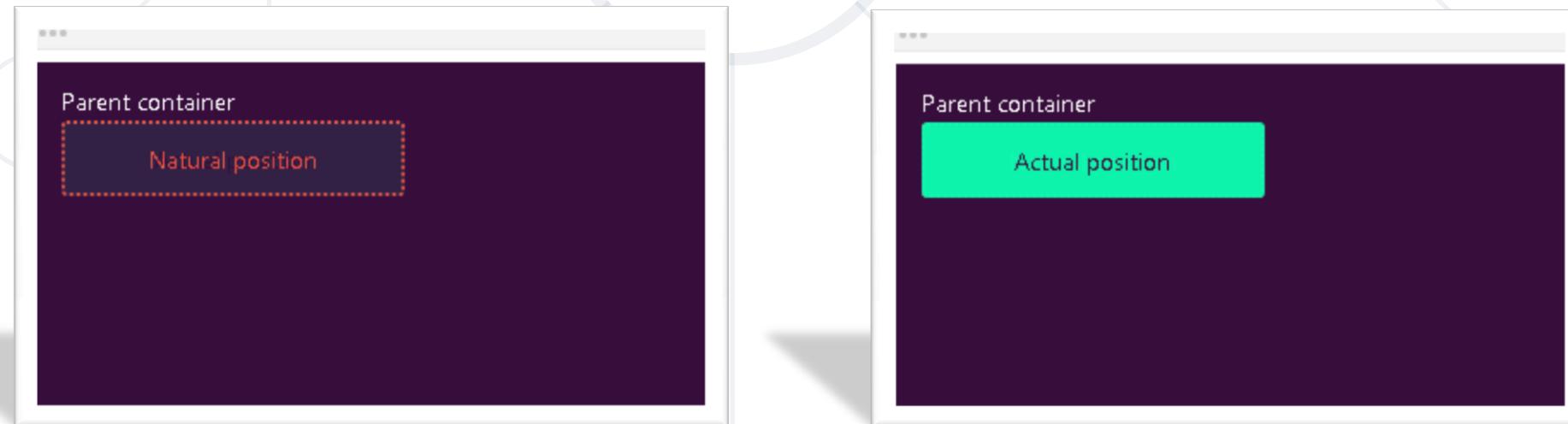
- `bottom: 20px;`
- If the element is in position **relative**, the element will move *upwards* by the amount defined by the **bottom** value



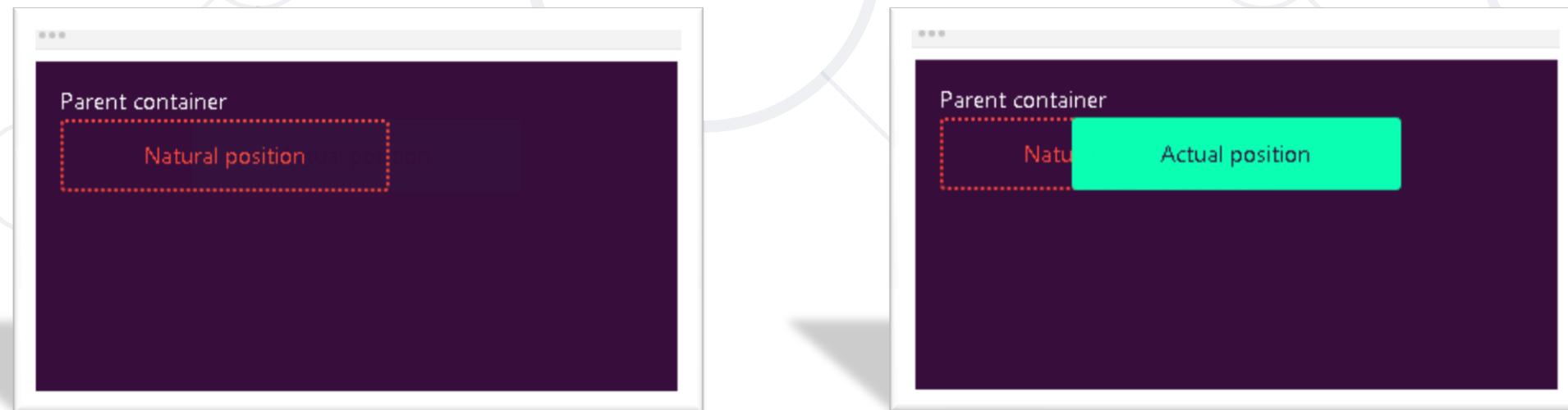
- `bottom: 0;`
- If the element is in position **absolute**, the element will position itself from the *bottom* of the first positioned **ancestor**



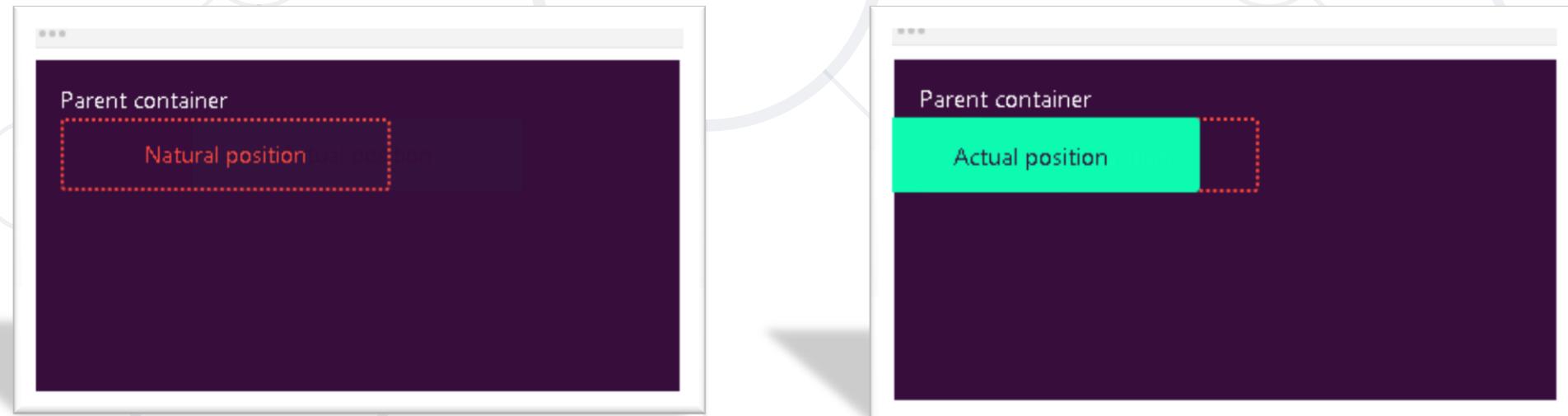
- Left – defines the position of the element according to its **left** edge
  - `left: auto;` - the element will remain in its **natural** position



- `left: 80px;` - if the element is in position **relative**, the element will move *left* by the amount defined by the **left** value

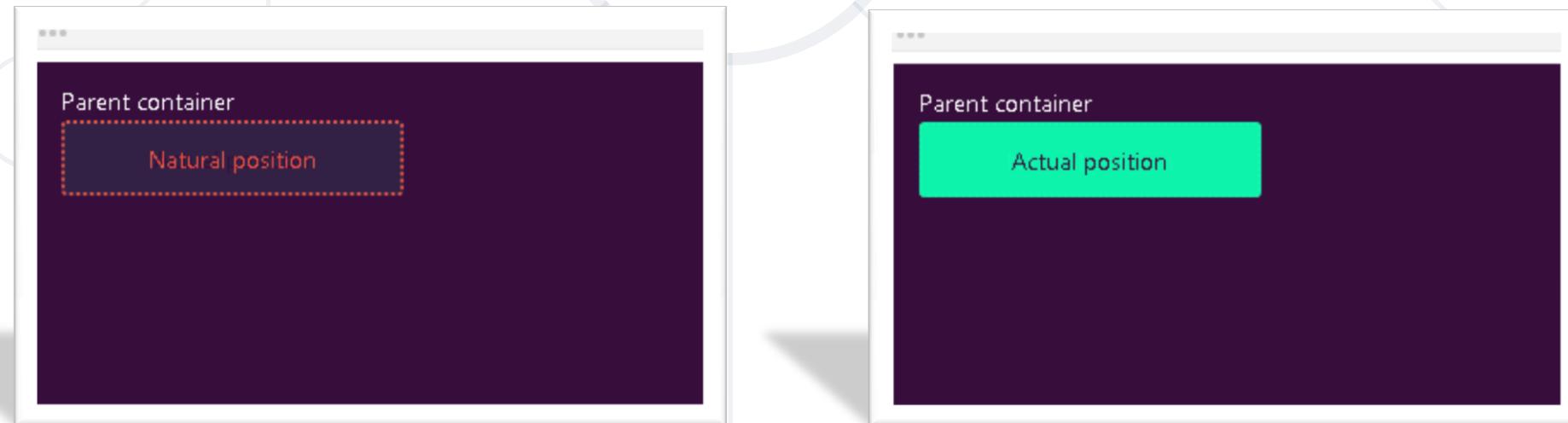


- `left: -20px;` - if the element is in position **absolute**, the element will position itself from the *left* of the first positioned **ancestor**



# Right

- Right – defines the position of the element according to its **right** edge
  - `right: auto;` - the element will remain in its **natural** position



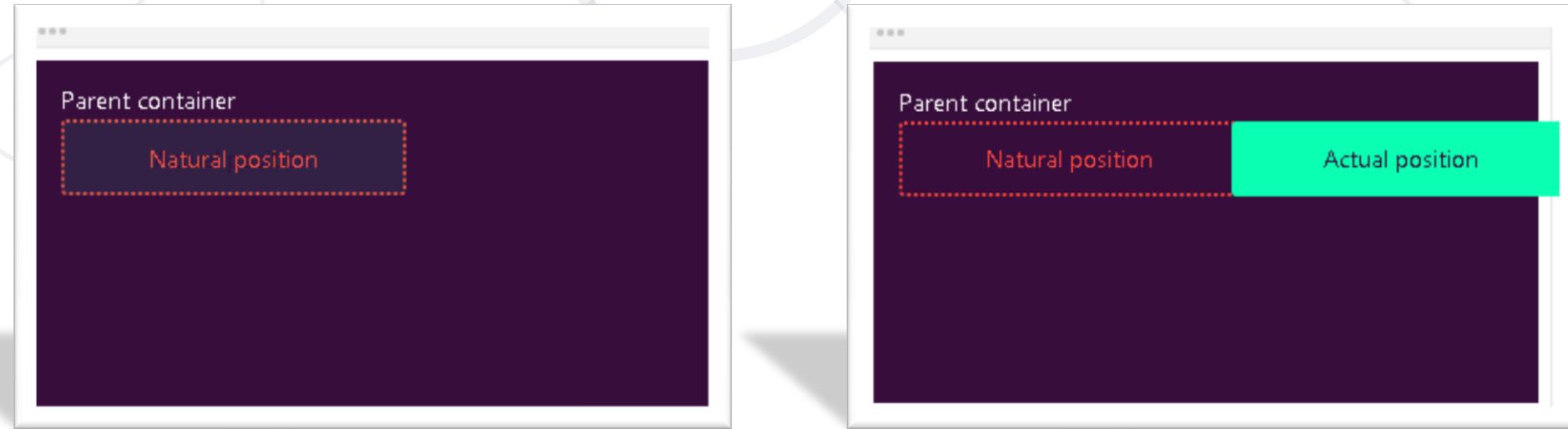
# Right

- `right: 80px;` - if the element is in position **relative**, the element will move *right* by the amount defined by the **right** value



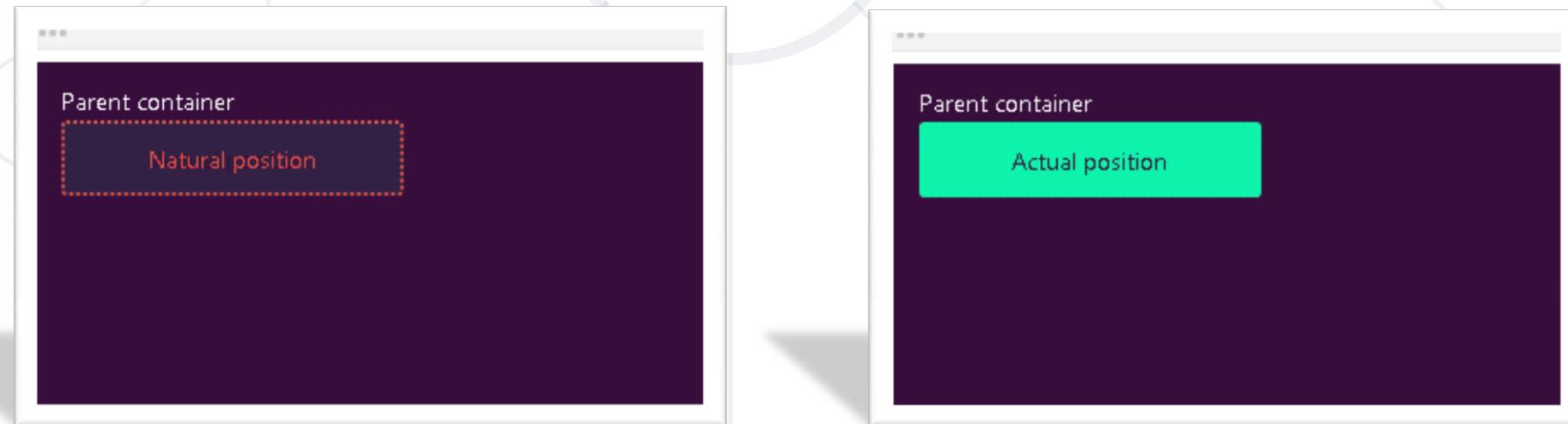
# Right

- `right: -20px;` - If the element is in position **absolute**, the element will position itself from the *right* of the first positioned **ancestor**

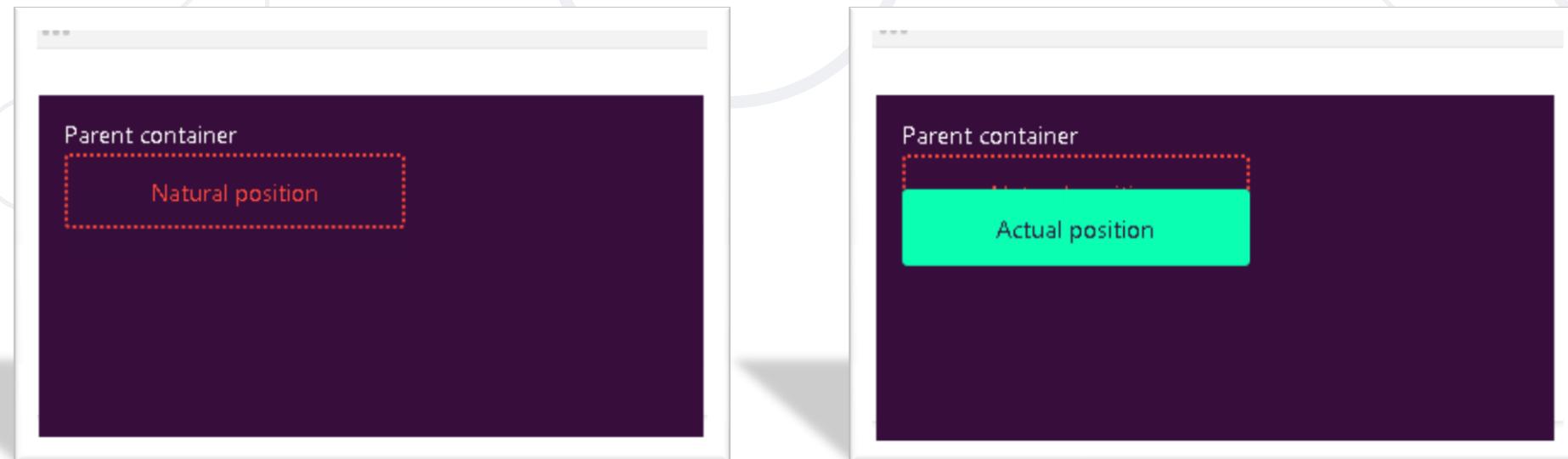


# Top

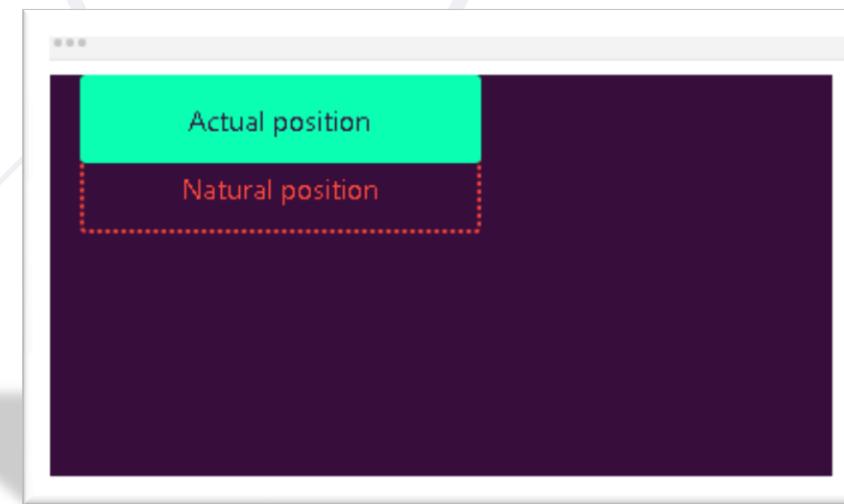
- Top – defines the position of the element according to its **top** edge
  - `top: auto;` - the element will remain in its **natural** position



- `top: 20px;` - If the element is in position **relative**, the element will move *downwards* by the amount defined by the **top** value



- `top: 0;` - If the element is in position **absolute**, the element will position itself from the *top* of the first positioned **ancestor**



- Defines the order of the elements on the **z-axis**. It only works on positioned elements (anything apart from static)
  - **z-index: auto;**
  - The order is defined by the order in the HTML code:
    - First in the code = behind
    - Last in the code = in front



- **z-index: 1;**
- The z-index value is relative to the other ones. The target element is move in **front** of its siblings



- **z-index: -1;**
- You can use negative values. The target element is move in behind its siblings



# Summary

- What is a float?
- How to float elements?
- How to clear float?
- Positioning properties
- How to work with z-index?



# SoftUni Diamond Partners



**xssoftware**



**SBTech**  
*we know sports*



**telenor**



**SoftwareGroup**  
*doing it right*

**NETPEAK**



**SmartIT**



**Postbank**

*Решения за твоето упре*

**SUPER  
HOSTING  
.BG**

**INDEAVR**  
*Serving the high achievers*

**INFRASTICS®**



**STEMO®**  
*Computer Systems & Software*



# SoftUni Organizational Partners



ИНФОРМАЦИОННО  
ОБСЛУЖВАНЕ

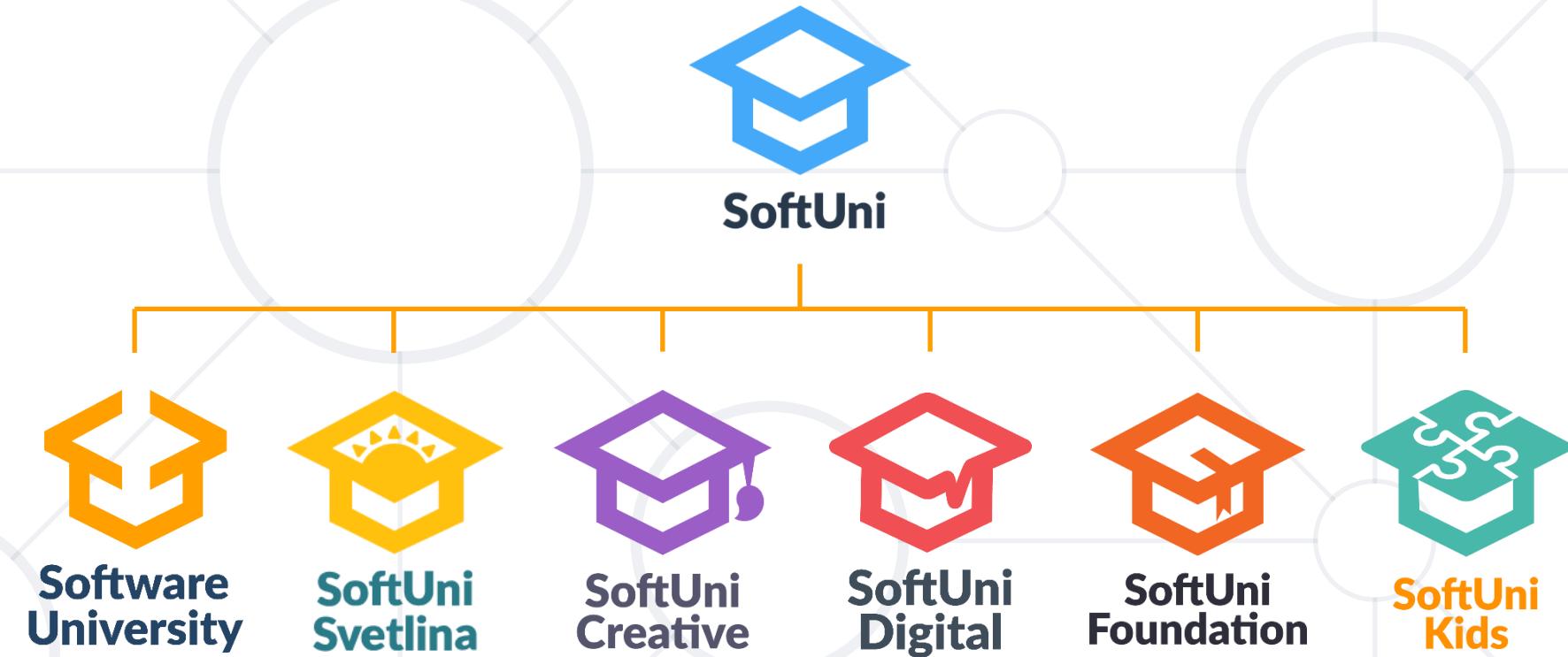
OneBit  
SOFTWARE



Lukanet.com



# Questions?



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>



# Trainings @ Software University (SoftUni)



- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Software University Forums
  - [forum.softuni.bg](http://forum.softuni.bg)



Software  
University



# FLEXBOX



SoftUni Team  
Technical Trainers



SoftUni



Software University  
<https://softuni.bg>

# Table of Contents

1. Flexbox
2. Properties for the Parent
3. Properties for the Children



Have a Question?

**sli.do**

**#HTML-CSS**

# FLEXBOX

*Flexbox*



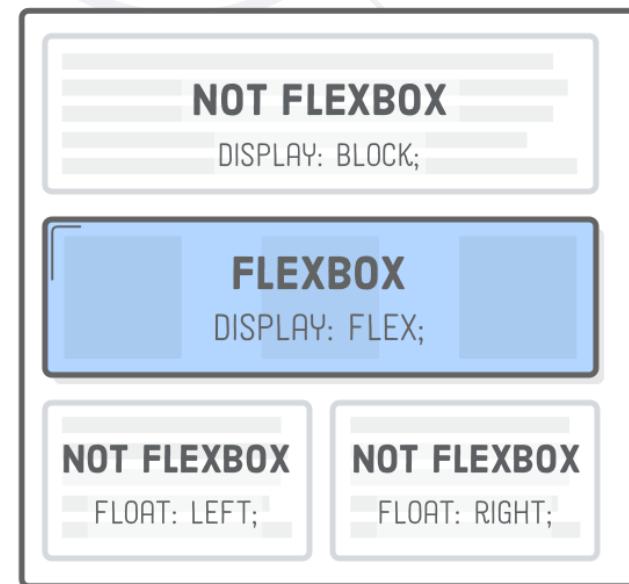
# What is Flexbox?

- The Flexible Box Module - **flexbox**, was designed as a one-dimensional layout model, and as a method that could offer **space distribution** between items in an interface and powerful alignment capabilities
- Flexbox is a method for laying out items in **rows** or **columns**
- Items flex to **fill** additional space and **shrink** to fit into smaller spaces



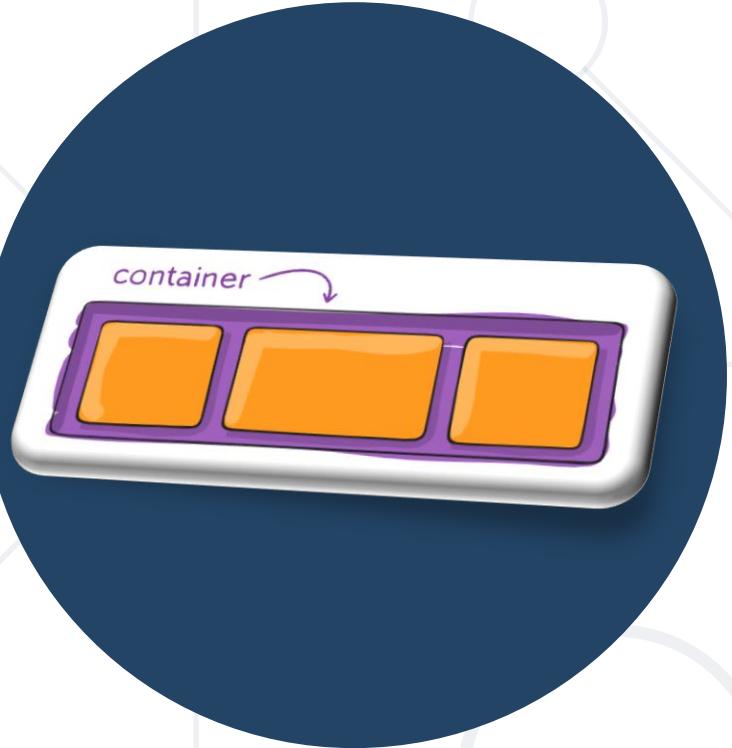
# Why Flexbox?

- For a long time, the only reliable cross browser-compatible tools available for creating CSS layouts were **floats** and **positioning**
- These are fine and they work, but in some ways, they are also rather limiting and frustrating



# Why Flexbox?

- The following simple layout requirements are either **difficult** or **impossible** to achieve with such tools, in any kind of convenient, flexible way:
  - Vertically centering a block of content inside its parent
  - Making all the children of a container take up an equal amount of the available width/height, regardless of how much width/height is available
  - Making all columns in a multiple column layout adopt the same height even if they contain a different amount of content



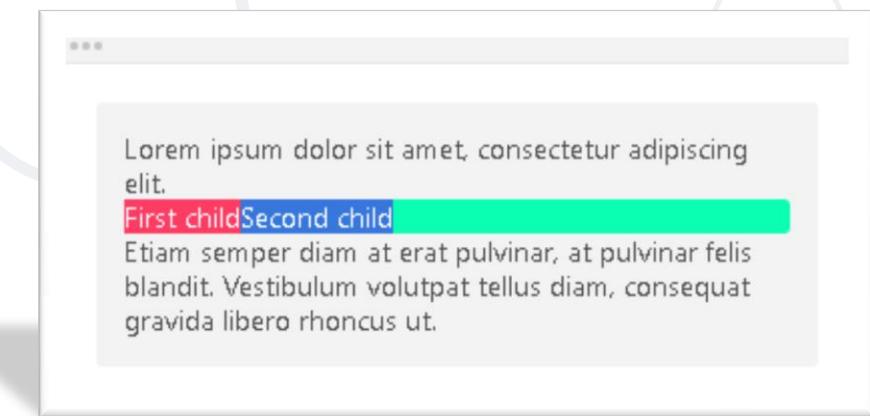
# Properties for the Parent (flex container)

# Display - flex

- The element is turned into a **flexbox** container. On its own, it behaves like a block element
- Its child elements will be turned into **flexbox items**

```
<body>
  <p>Lorem ipsum dolor sit amet, co
nsectetur adipiscing elit.</p>
  <div class="container">
    <p>First child</p>
    <p>Second child</p>
  </div>
  <p>Etiam semper diam at erat pulv
inar, at pulvinar felis blandit. Vest
ibulum volutpat tellus diam, consequa
t gravida libero rhoncus ut.
  </p>
</body>
```

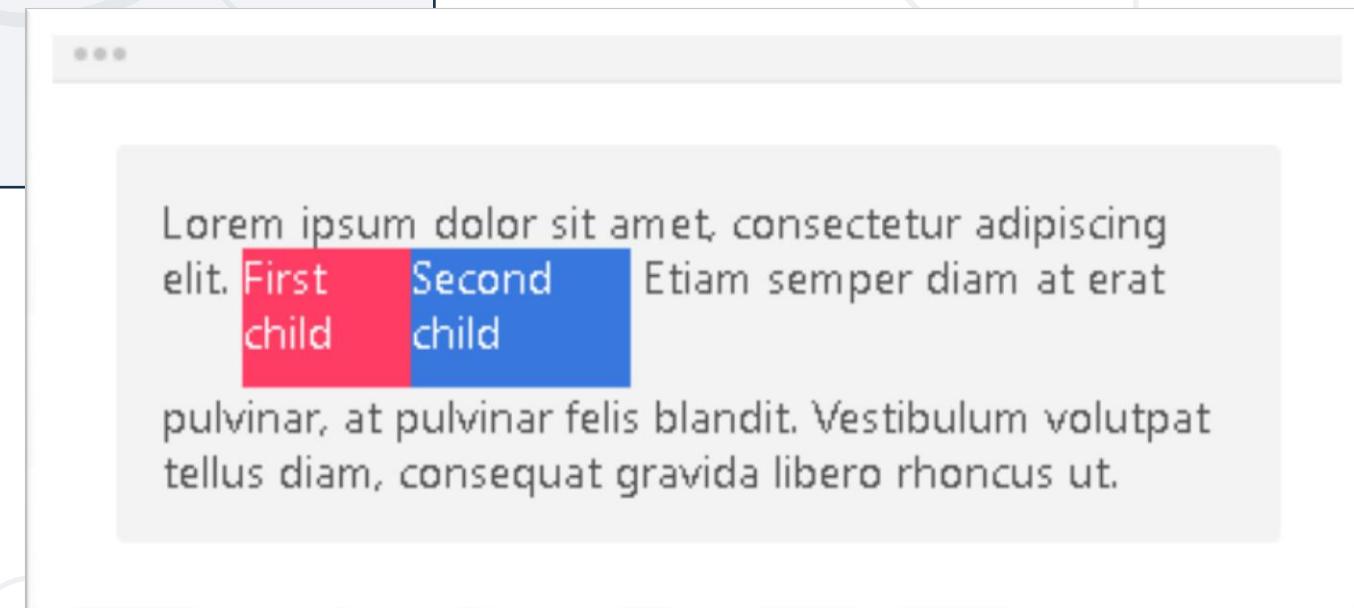
```
.container {
  display: flex;
}
```



- The element shares properties of both an **inline** and a **flexbox** element:
  - **inline** because the element behaves like simple text, and inserts itself in a block of text
  - **flexbox** because its child element will be turned into flexbox items

# Display – inline-flex Example

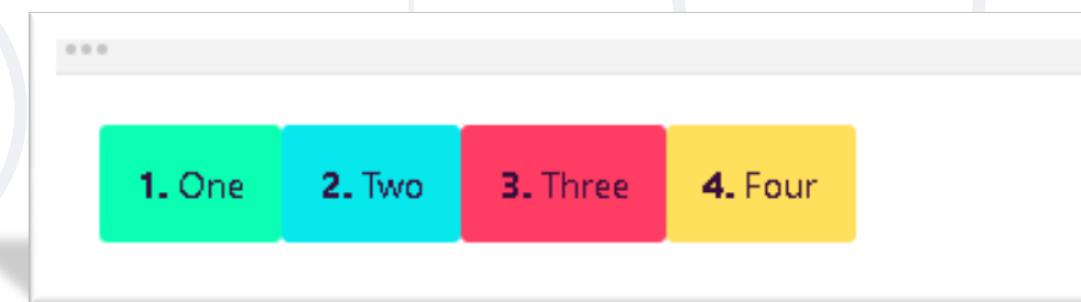
```
.container {  
    display: inline-flex;  
    height: 3em;  
    width: 120px;  
}
```



# Flex-direction

- Defines how flexbox items are ordered within a flexbox container
  - **flex-direction: row;**

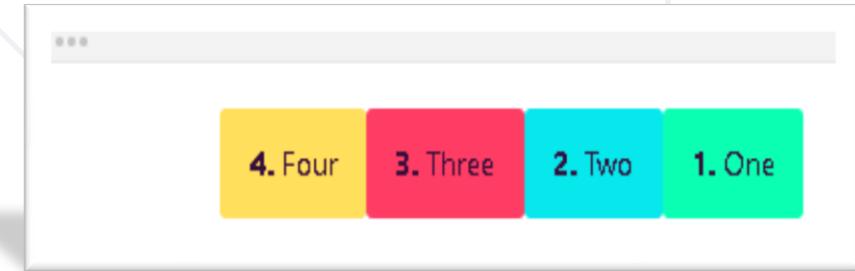
The flexbox items are ordered the **same way** as the text direction, along the main axis



# Flex-direction

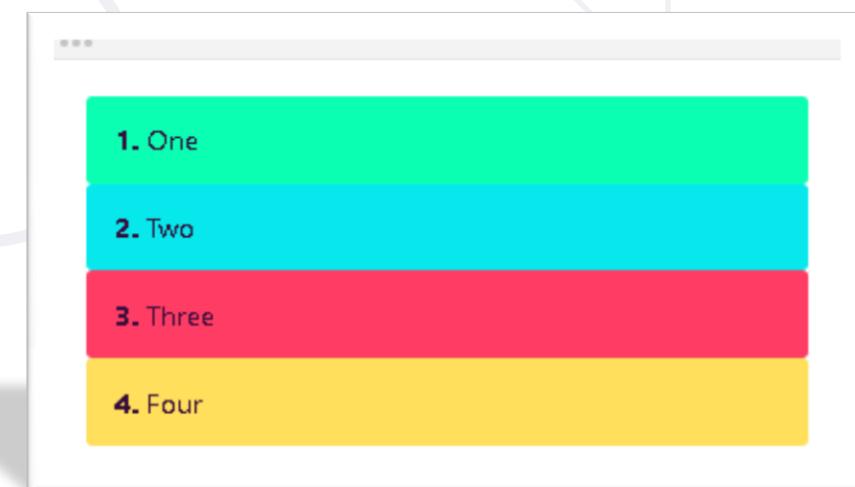
- **flex-direction: row-reverse;**

The flexbox items are ordered the **opposite** way as the **text direction**, along the main axis



- **flex-direction: column;**

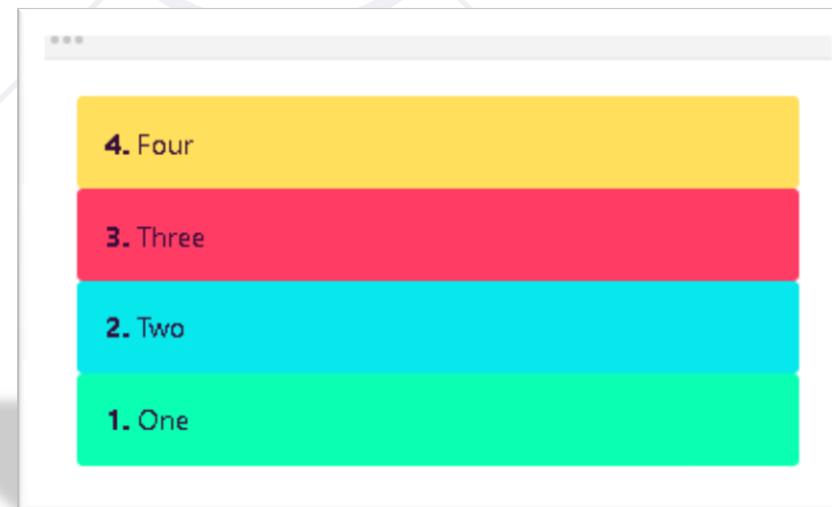
The flexbox items are ordered the **same** way as the **text direction**, along the **cross axis**



# Flex-direction

- **flex-direction: column-reverse;**

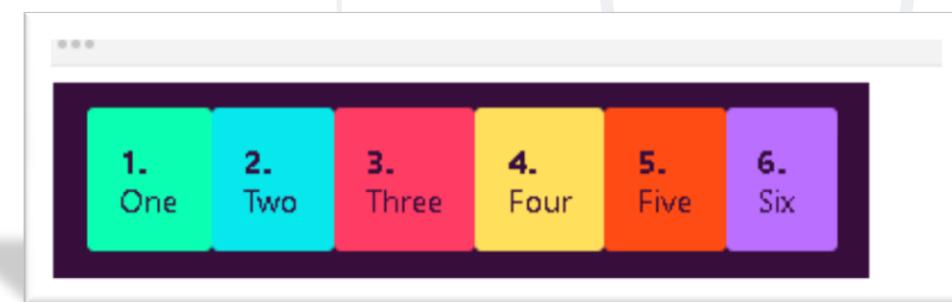
The flexbox items are ordered the **opposite way** as the **text direction**, along the **cross axis**



# Flex-wrap

- Defines if flexbox items appear on a **single line** or on **multiple lines** within a flexbox container
  - **flex-wrap: nowrap;**

The flexbox items will remain on a **single line**, no matter what, and will eventually overflow if needed



# Flex-wrap

- **flex-wrap: wrap;**

The flexbox items will be distributed among **multiple lines** if needed



- **flex-wrap: wrap-reverse;**

The flexbox items will be distributed among **multiple lines** if needed. Any additional line will appear **before** the previous one



- **Flex-flow** is a shorthand for the flex-direction and flex-wrap properties
- The default value is **row nowrap**

```
flex-flow: <flex-direction> || <flex-wrap>

.container {
  flex-flow: row wrap;
}
```

# Justify-content

- Justify-content: Defines how flexbox/grid items are aligned according to the **main** axis, within a flexbox container
  - **justify-content: flex-start;**The flexbox items are pushed towards the **start** of the container's main axis



# Justify-content

- **justify-content: flex-end;**

The flexbox items are pushed towards the **end** of the container's main axis



- **justify-content: center;**

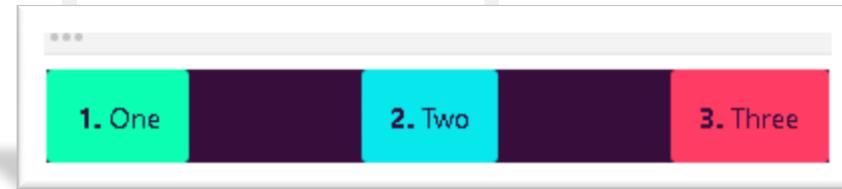
The flexbox items are **centered** along the container's main axis



# Justify-content

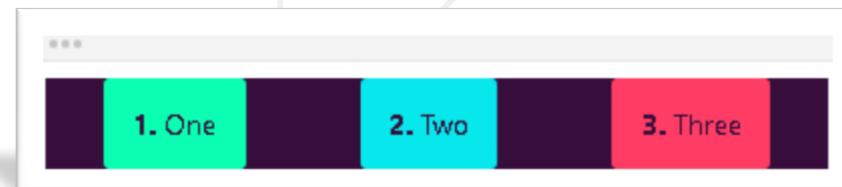
- **justify-content: space-between;**

The remaining space is distributed **between** the flexbox items



- **justify-content: space-around;**

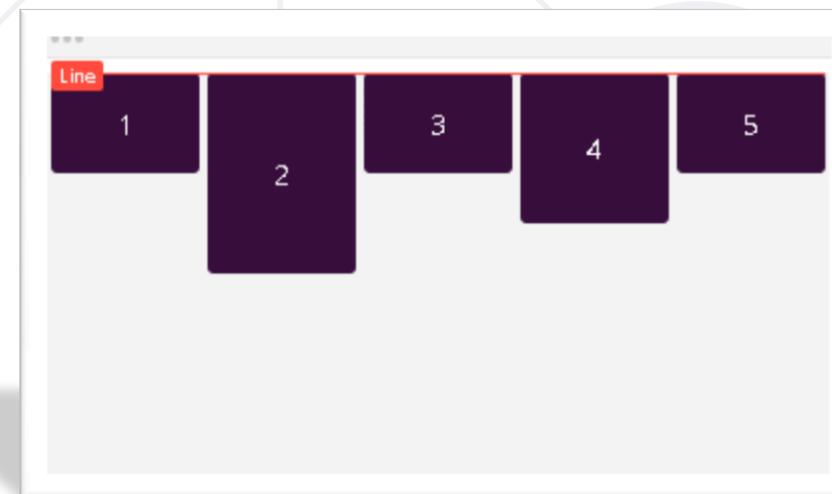
The remaining space is distributed **around** the flexbox items: this adds space **before** the first item and **after** the last one



# Align-items

- Align-items: Defines how flexbox items are aligned according to the **cross axis**, within a line of a flexbox container
  - **align-items: flex-start;**

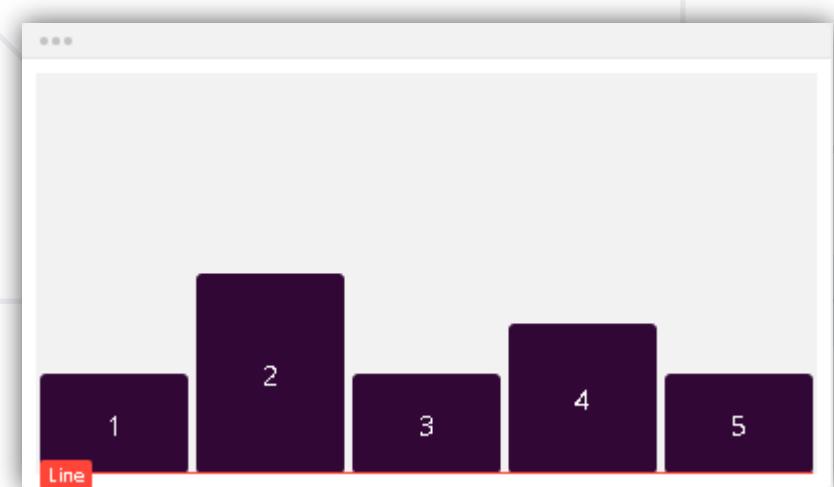
The flexbox items are aligned at the **start** of the **cross axis**



# Align-items

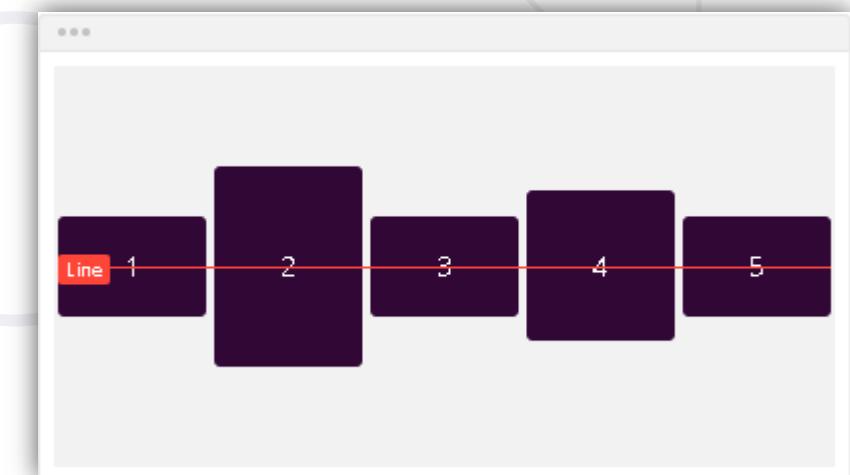
- **align-items: flex-end;**

The flexbox items are aligned at the **end** of the **cross axis**



- **align-items: center;**

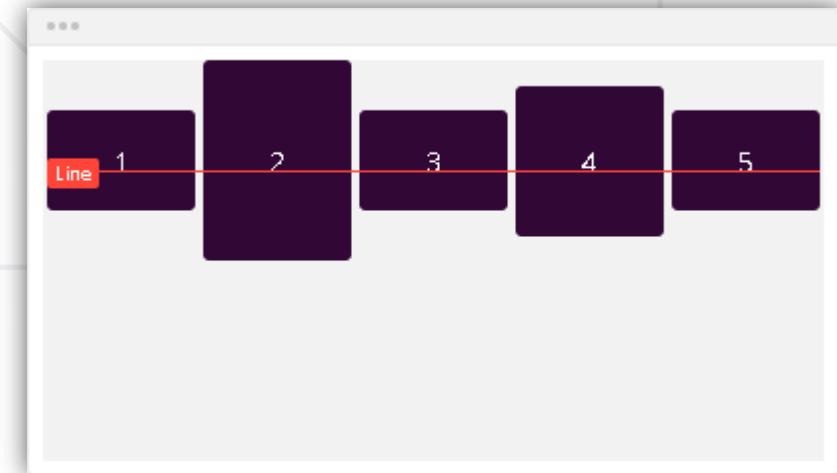
The flexbox items are aligned at the **center** of the **cross axis**



# Align-items

- **align-items: baseline;**

The flexbox items are aligned at the **baseline** of the **cross axis**



- **align-items: stretch;**

The flexbox items will stretch across the whole **cross axis**



# Align-content

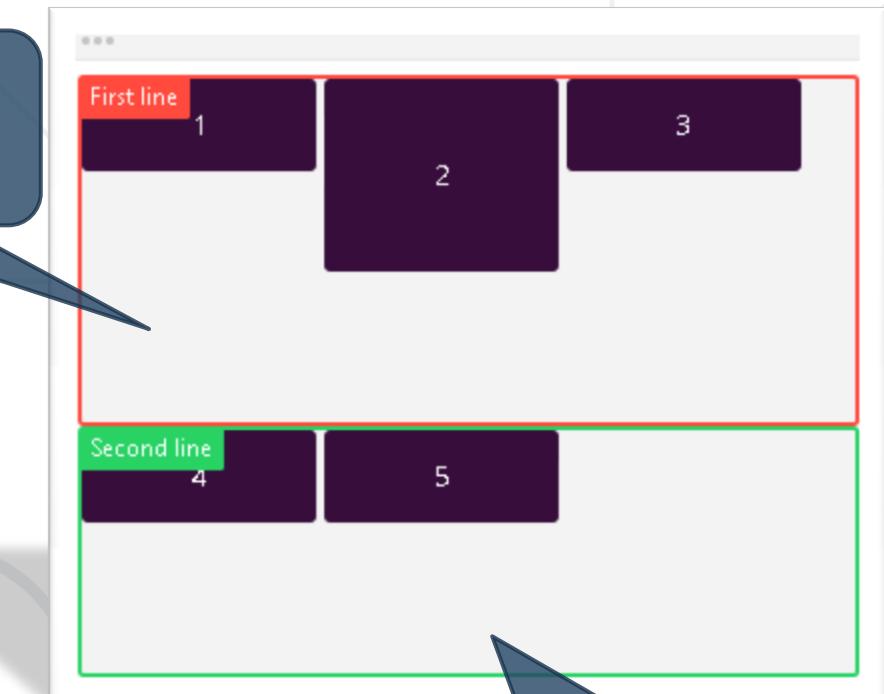
- Align-content: Defines how each line is aligned within a flexbox container
- It only applies if flex-wrap: wrap is present, and if there are multiple lines of flexbox items
  - **align-content: stretch;**  
Each line will stretch to **fill** the remaining space

# Align-content: stretch Example

- Example:

The container is 300px high  
All boxes are 50px high  
The second box is 100px high

The first line is  
175px high



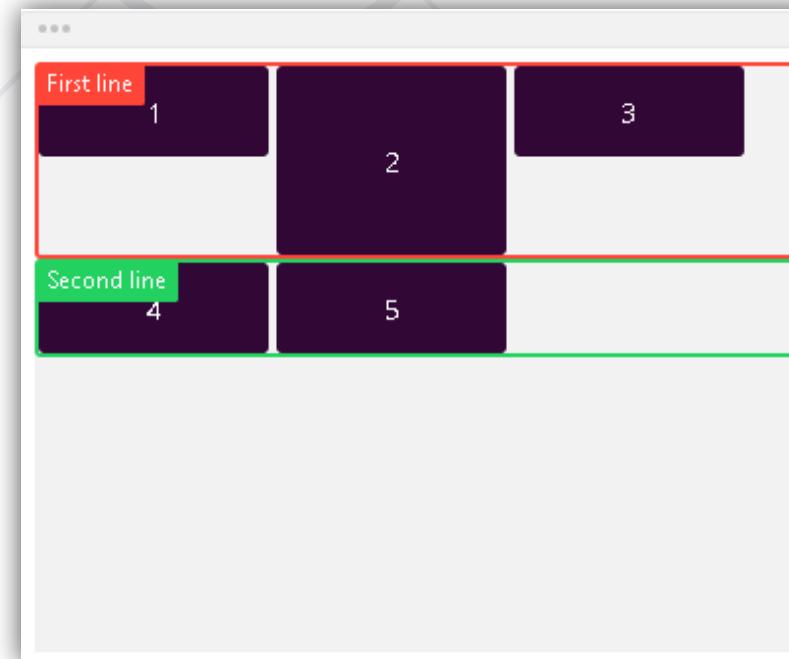
- The first line is 100px high
- The second line is 50px high
- The remaining space is 150px and it is distributed equally amongst the two lines

The second line is  
125px high

# Align-content

- **align-content: flex-start;**

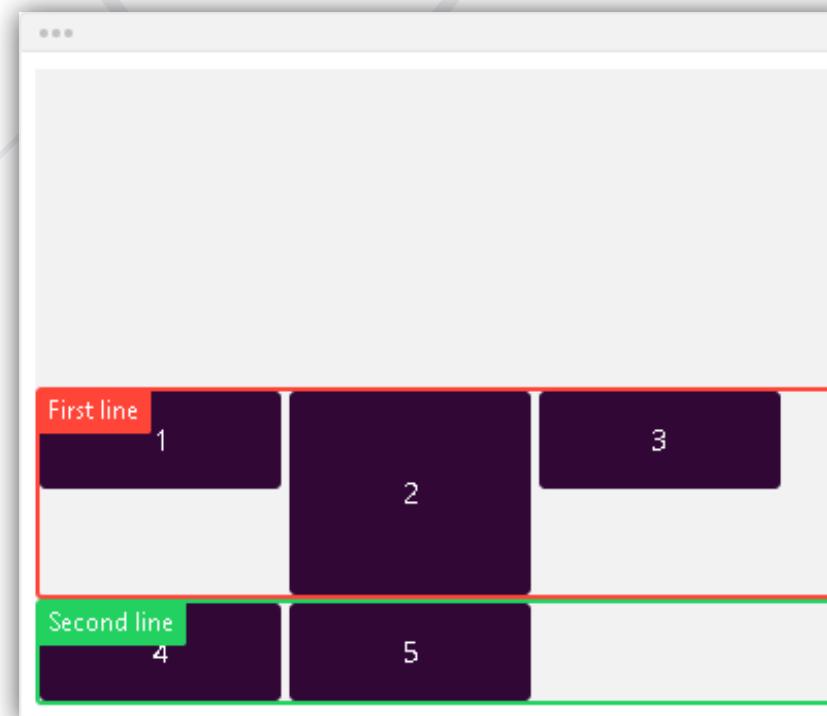
Each line will only fill the space it *needs*. They will all move towards the **start** of the flexbox container's cross axis



# Align-content

- **align-content: flex-end;**

Each line will only fill the space it *needs*. They will all move towards the **end** of the flexbox container's cross axis



# Align-content

- **align-content: center;**

Each line will only fill the space it *needs*. They will all move towards the **center** of the flexbox container's cross axis



# Align-content

- **align-content: space-between;**

Each line will only fill the space it *needs*. The *remaining* space will appear **between** the lines



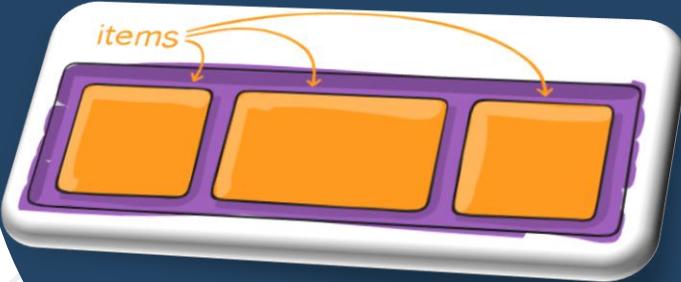
# Align-content

- **align-content: space-around;**

Each line will only fill the space it *needs*. The **remaining** space will be distributed equally **around** the lines: before the first line, between the two, and after the last one



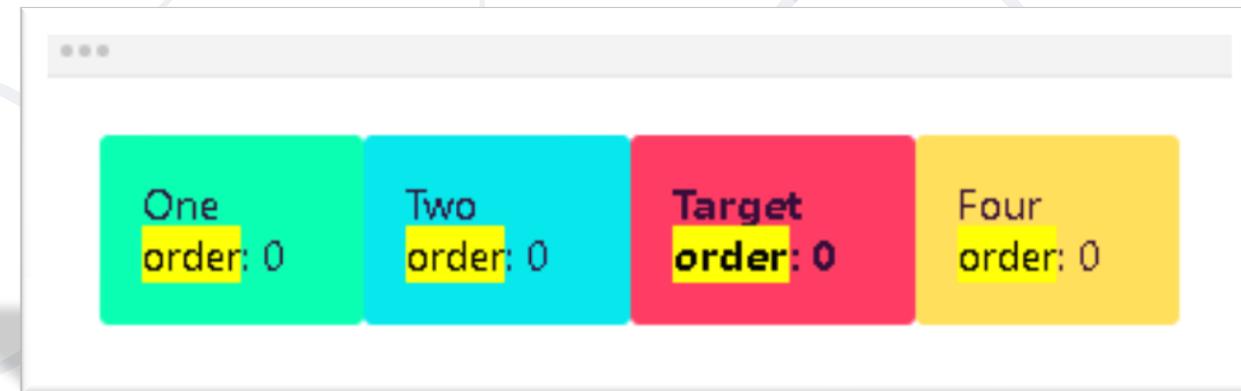
# Properties for the Children (flex items)



- Order - defines the order of a flexbox item

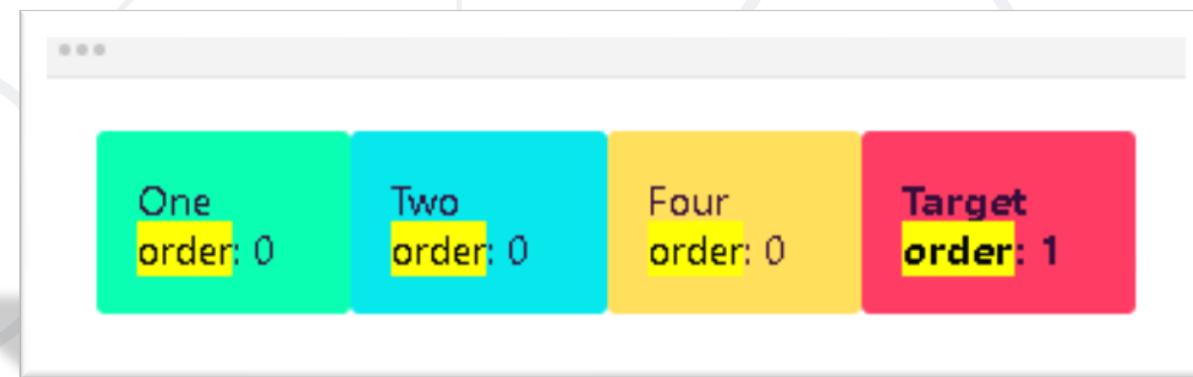
- **order: 0;**

The order of the flexbox items is the one defined in the **HTML code**



- **order: 1;**

The order is **relative** to the flexbox item's **siblings**. The final order is defined when all individual flexbox item order values are taken into account

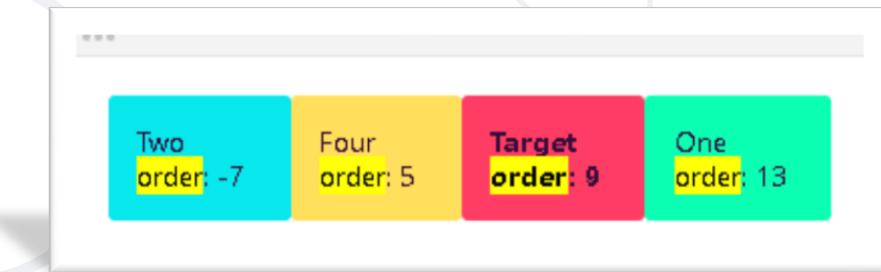
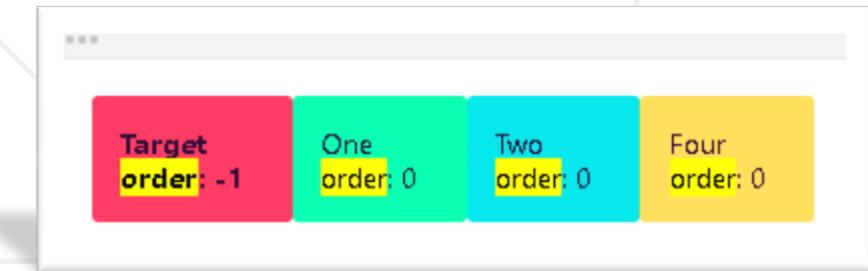


- **order: -1;**

You can use **negative values**

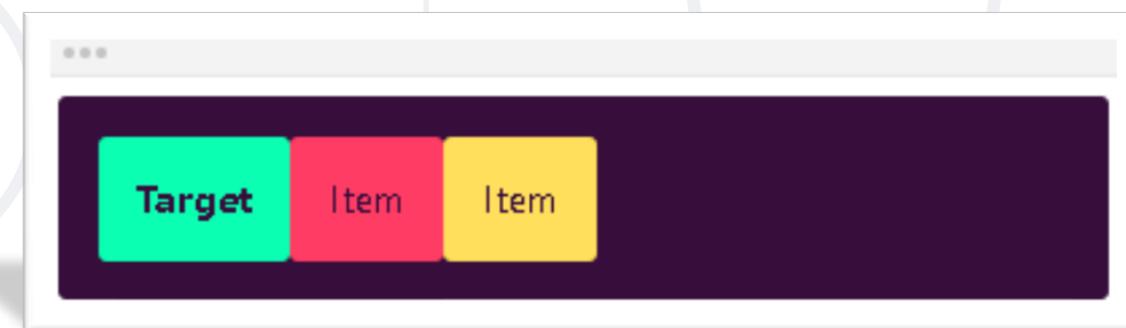
- **order: 9;**

You can set a **different** value for each flexbox item



# Flex-grow

- Flex-grow - defines how much a flexbox item should **grow** if there's space available
  - **flex-grow: 0;**The element will **not** grow if there's space available. It will only use the space it needs



# Flex-grow

- **flex-grow: 1;**

The element will **grow** by a factor of 1. It will fill up the remaining space if no other flexbox item has a flex-grow value



# Flex-shrink

- Flex-shrink - defines how much a flexbox item should **shrink** if there's **not enough** space available

- **flex-shrink: 1;**

If there's **not enough** space available in the container's main axis, the element will **shrink** by a factor of **1**, and will wrap its content



- **flex-shrink: 0;**

The element will **not** shrink it will retain the width it needs, and **not** wrap its content. Its siblings will shrink to give space to the target element.

Because the target element will not wrap its content, there is a chance for the flexbox container's content to **overflow**

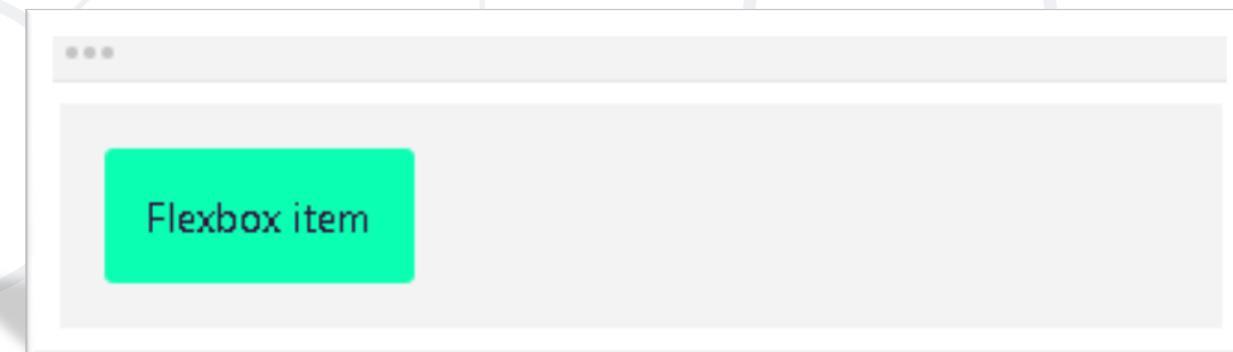


# Flex-basis

- Flex-basis - defines the initial size of a flexbox item

- **flex-basis: auto;**

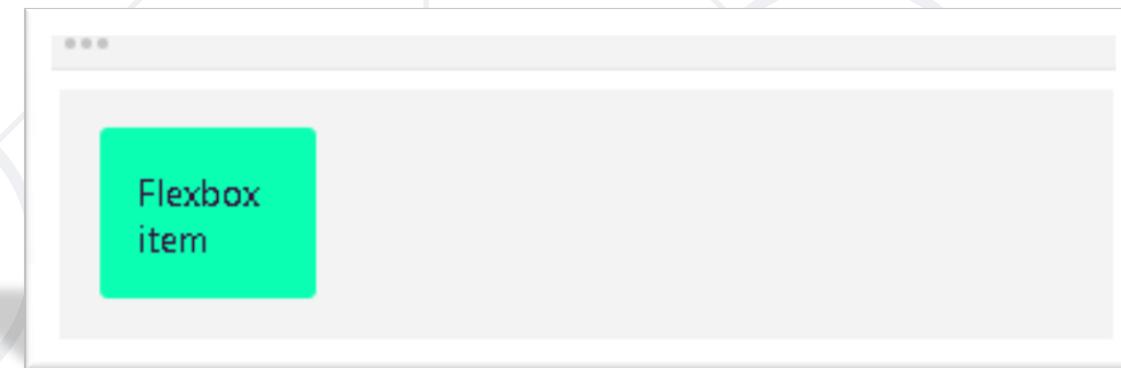
The element will be automatically sized based on its content, or on any height or width value if they are defined



# Flex-basis

- **flex-basis: 80px;**

You can define **pixel** or **(r)em** values. The element will wrap its content to avoid any overflow



- Flex is the shorthand for:
  - flex-grow
  - flex-shrink
  - flex-basis
- The default is **0 1 auto**

```
.item {  
  flex: <flex-grow> <flex-shrink> <flex-basis>  
}
```

# Align-self

- Align-self – works like align-items, but applies only to a **single** flexbox item, instead of all of them

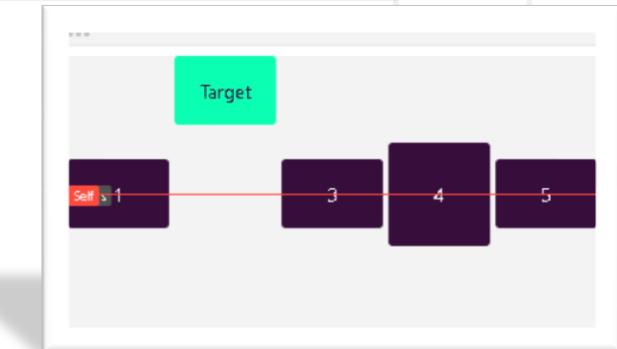
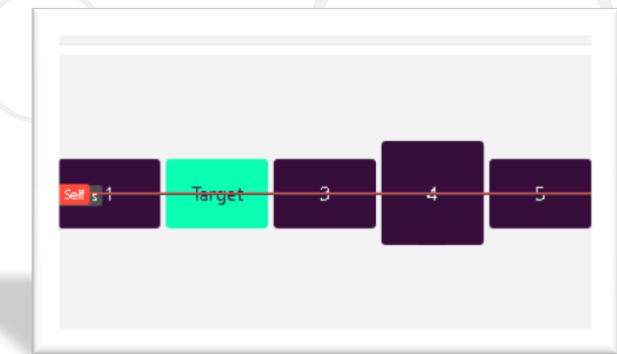
- **align-self: auto;**

The target will use the value of align-items

- **align-self: flex-start;**

The container has align-items: center

The target has align-self: flex-start



# Align-self

- **align-self: center;**

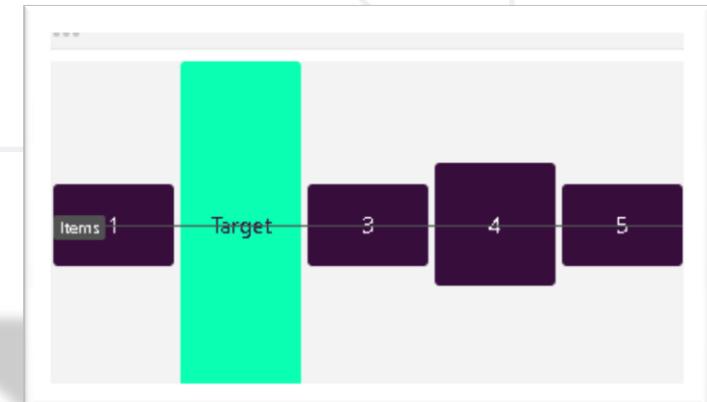
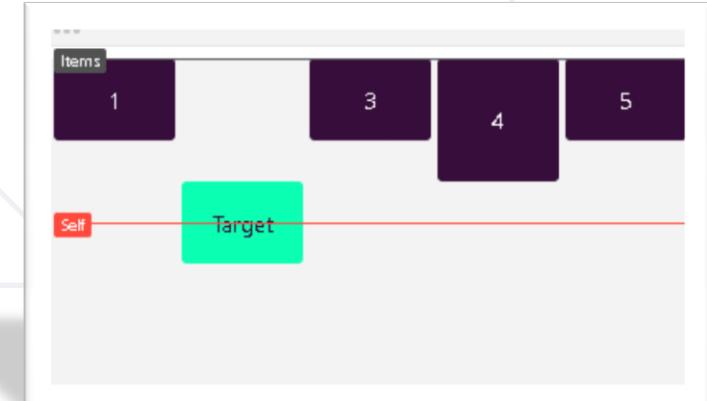
The container has align-items: flex-start

The target has align-self: center

- **align-self: stretch;**

The container has align-items: center

The target has align-self: stretch



- **What is Flexbox?**
- **Why Flexbox?**
- **Properties for the Parent: display, direction, wrap, justify, align**
- **Properties for the children: order, shrink, align**



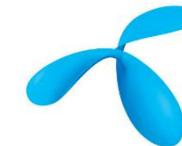
# SoftUni Diamond Partners



**XS**software



**SBTech**  
*we know sports*



telenor



**SoftwareGroup**  
*doing it right*

**NETPEAK**



**SmartIT**



**Postbank**

*Решения за твоето упре*

**SUPER  
HOSTING**  
:BG

**INDEAVR**  
*Serving the high achievers*

**INFRASTICS®**



**STEMO®**  
*Computer Systems & Software*



# SoftUni Organizational Partners



ИНФОРМАЦИОННО  
ОБСЛУЖВАНЕ

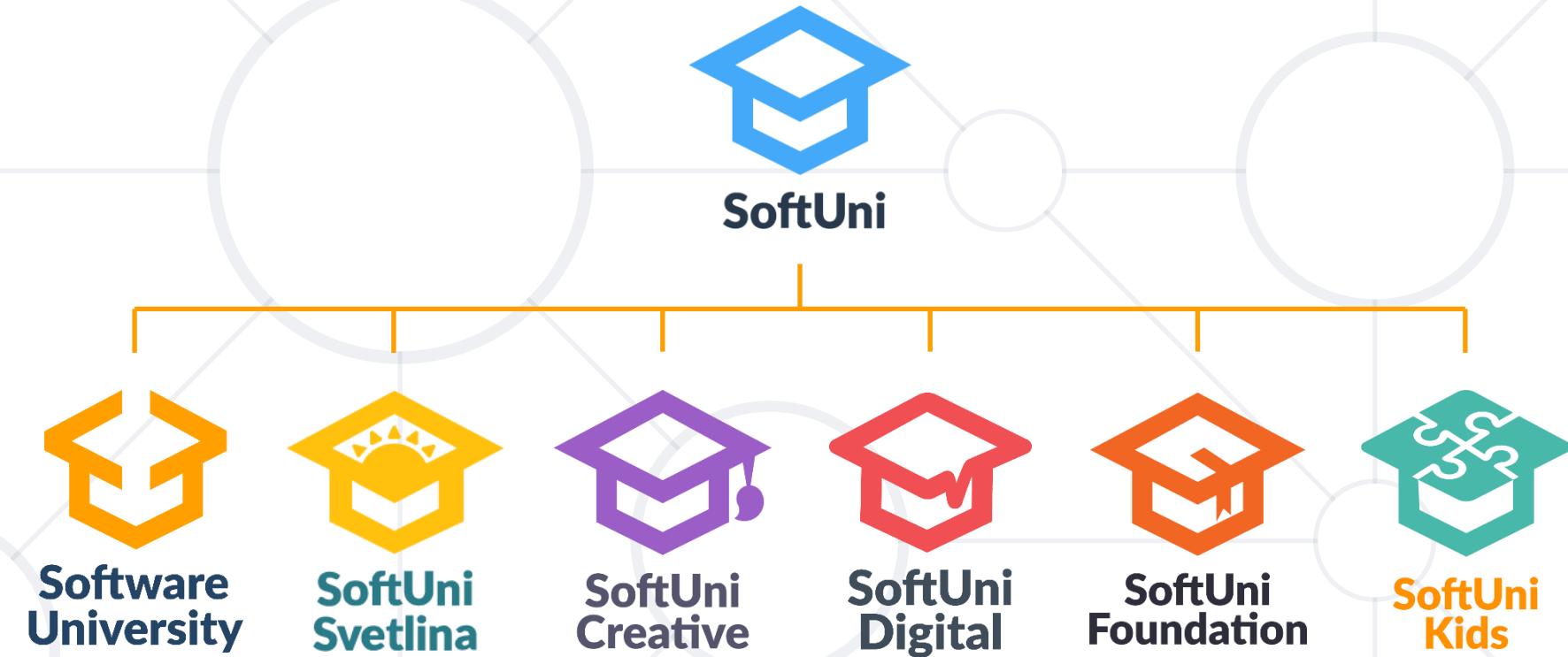
OneBit  
SOFTWARE



Lukanet.com



# Questions?



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>



# Trainings @ Software University (SoftUni)



- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Software University Forums
  - [forum.softuni.bg](http://forum.softuni.bg)



Software  
University



# MEDIA QUERIES



SoftUni Team  
Technical Trainers



SoftUni



Software University  
<https://softuni.bg>

# Table of Contents

1. Responsive Web Design
2. Media Queries
3. Media Types
4. Media Feature Rules
5. Logical Operators
6. @import



Have a Question?



sli.do

#HTML-CSS

# Responsive Web Design



# What is Responsive Web Design?

- Responsive Web Design is about using HTML and CSS to **automatically resize**, hide, shrink, or enlarge, a website, to make it look good on **all devices** (desktops, tablets, and phones)
- Setting the Viewport – add the following <meta> element:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- This will give the browser instructions on how to control the page's **dimensions** and **scaling**

# Components of Responsive Design

- Responsive website design consists of the following three main components:
  - **Flexible layouts** – Using a flexible grid to create the website layout. That will dynamically resize to any width
  - **Media queries** - Allow designers to specify different styles for specific browser and device circumstances
  - **Flexible Media** – Makes media (images, video and other format) scalable

# Benefits of using a Responsive Website

- Increased **traffic** from mobile users
- Lower **cost** and website maintenance
- Provides a Seamless **User Experience**
- Adapts Easily To Any Screen Size
- A Responsive Website Improves Your **SEO** Efforts

# Why Responsive Website Design Works?

- Google Prioritizes Responsive Websites
- 50% of Total eCommerce Revenue Comes from Mobile
- 94% of People Judge Websites on Responsive Web Design
- Almost 60% of All Internet Access is Done Through the Phone
- 77% of Adults Own a Smart Phone
- 72% of People Want Mobile-Friendly Websites
- Responsive Design Integrates Social Media

# What is a Media Query?

- Media Queries are a feature of CSS that enable webpage content to **adapt** to different screen sizes and resolutions
- They are a fundamental part of **responsive web design** and are used to customize the appearance of websites for multiple devices



- Media queries in CSS3 look at the **capability** of the device
- Media queries can be used to check many things, such as:
  - width and height of the viewport
  - width and height of the device
  - orientation (is the tablet/phone in landscape or portrait mode?)
  - resolution

# Media Query Syntax

- A media query consists of a **media type** and can contain one or more **expressions**, which resolve to either true or false

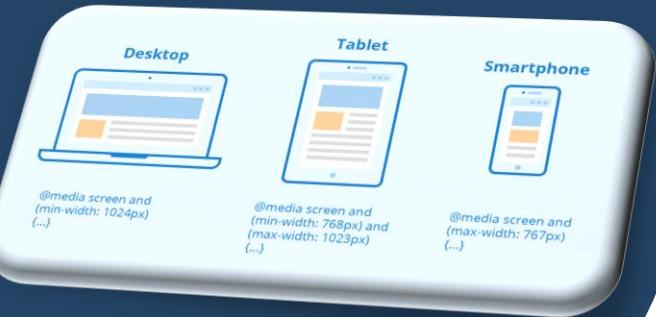
```
@media not|only mediatype and (media-feature-rule) {  
    CSS rules goes here  
}
```

- The result of the query is **true** if the specified media type matches the type of device the document is being displayed on
- Unless you use the **not** or **only** operators, the media type is **optional** and the all type will be implied

# Media Query Syntax

- A **media type**, which tells the browser what kind of media this code is for (e.g. print, or screen)
- A **media feature rule** - test that must be passed for the contained CSS to be applied
- A set of **CSS rules** that will be applied if the test passes and the media type is correct

# Media Types



- Media Types describe the general category of a given device:
  - **All** – used for all media type devices
  - **Print** – used for printers
  - **Screen** – used for computer screens, tablets, smart-phones etc.
  - **Speech** – used for screenreaders that "reads" the page out loud

# Media Query - Example

- The following media query will only set the body to 12px if the page is **printed**. It will not apply when the page is loaded in a browser:

```
@media print {  
    body {  
        font-size: 12px;  
    }  
}
```

# Media Feature Rules

## MEDIA FEATURE

`@media and/not/only (media feature)  
height  
width  
max-height / min-height  
max-width / min-width`

# Media Feature Rules (1)

- After specifying the type, you can then target a media feature with a **rule**
  - Width and height – we can apply CSS if the viewport is above or below a certain width, using **width**

```
@media screen and (width: 600px) {  
    body {  
        color: red;  
    }  
}
```

# Media Feature Rules (2)

- We can apply CSS if the viewport is with an exact width – using **min-width, max-width**

```
@media screen and (max-width:400px) {  
    body {  
        color: blue;  
    }  
}
```

# Media Feature Rules (3)

- Orientation – allows to test for **portrait** or **landscape** mode
- To change the body text color if the device is in landscape orientation:

```
@media (orientation: landscape) {  
    body {  
        color: red;  
    }  
}
```

# Media Feature Rules (4)

- Hover: used to query the user's ability to **hover** over elements on the page with the primary pointing device
  - `hover: over` elements with ease

```
@media (hover: hover) {  
    /* ... */  
}
```

- `none`: can't hover over elements

```
@media (hover: none) {  
    /* ... */  
}
```

# Media Feature Rules (5)

- Pointer: used to query the presence and accuracy of a **pointing** device such as a mouse
- If the primary input mechanism of the device includes:
  - a pointing device of limited accuracy, we use **coarse**

# Media Feature Rules (6)

- an accurate pointing device, we use **fine**
- **not include** a pointing device, we use **none**

```
/* define styles based on input device pointer accuracy */
@media(pointer: <course| fine | none>) {
    /* ... */
}
```

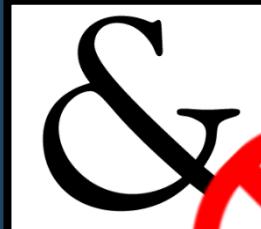
# Media Feature Rules (7)

- Target specific existing device

```
/* smartphones, touchscreens */
@media (hover: none) and (pointer: coarse) {
/* ... */
}
```

```
/* stylus-based screens */
@media (hover: none) and (pointer: fine) {
/* ... */
}
```

```
/* mouse, touch pad */
@media (hover: hover) and (pointer: fine) {
/* ... */
}
```



Only

# Logical Operators

# Logical Operators (1)

- The logical operators **not**, **and**, and **only** can be used to compose a complex media query
  - AND** - combining multiple media features

```
@media screen and (min-width: 400px) and (orientation: landscape) {  
    body {  
        color: blue;  
    }  
}
```

# Logical Operators (2)

- **NOT** - negate a media query

```
@media not all and (orientation: landscape) {  
    body {  
        color: blue;  
    }  
}
```

- **ONLY** - used to apply a style only if an entire query matches
- **,(comma)** - commas are used to combine multiple media queries into a single rule



*@import*

**@import**

- The **@import CSS at-rule** is used to import style rules from other style sheets
- You can specify media-dependent @import rules
- These conditional imports specify comma-separated media queries after the URL

```
@import url;  
@import url list-of-media-queries;  
@import url("fineprint.css") print;  
@import url("landscape.css") screen and (orientation: landscape);
```

- **What is Responsive Web Design?**
- **What are Media Queries?**
- **Media Types**
- **Media Feature Rules**
- **Logical Operators**
- **How to use @import?**



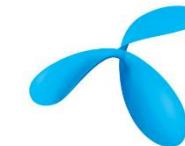
# SoftUni Diamond Partners



**xssoftware**



**SBTech**  
*we know sports*



**telenor**



**SoftwareGroup**  
*doing it right*

**NETPEAK**



**SmartIT**



**Postbank**

*Решения за твоето упре*

**SUPER  
HOSTING  
.BG**

**INDEAVR**  
*Serving the high achievers*

**INFRASTICS®**



**STEMO®**  
*Computer Systems & Software*



# SoftUni Organizational Partners



ИНФОРМАЦИОННО  
ОБСЛУЖВАНЕ

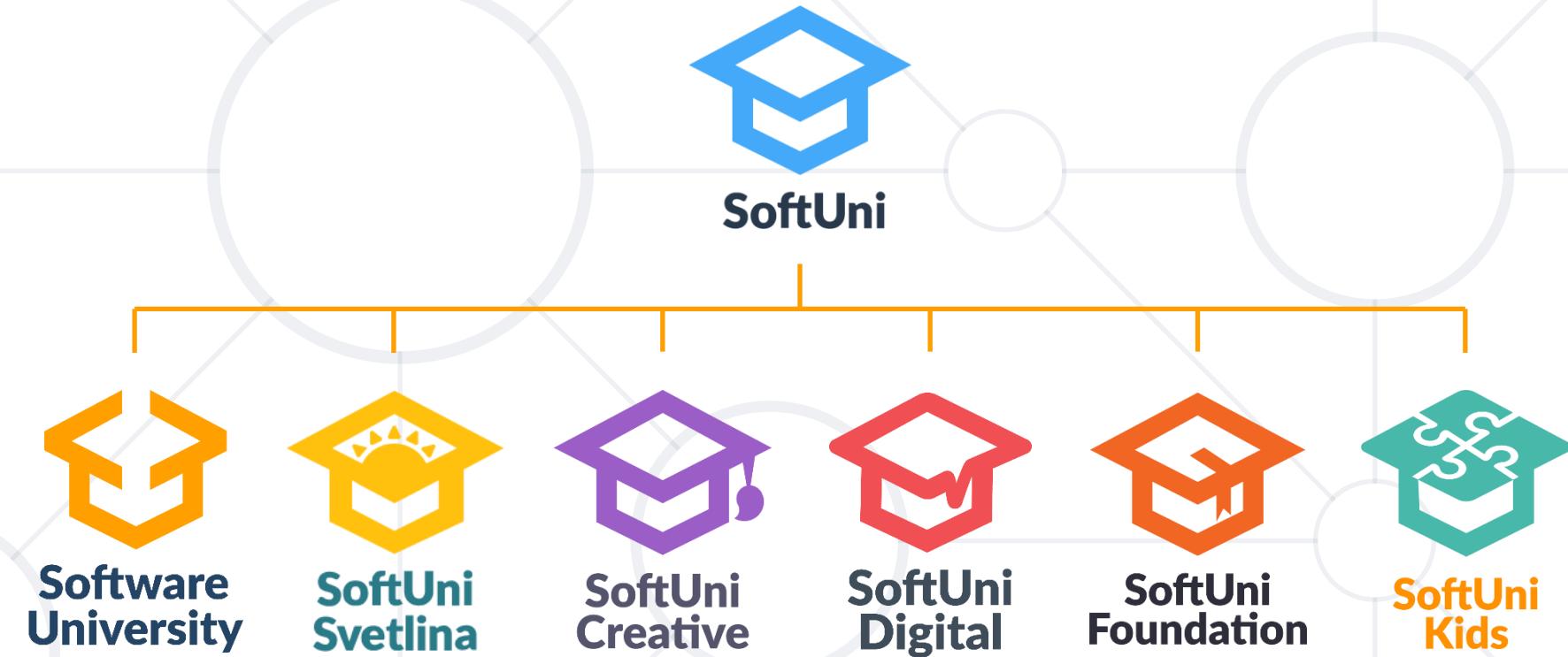
OneBit  
SOFTWARE



Lukanet.com



# Questions?



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>



# Trainings @ Software University (SoftUni)



- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Software University Forums
  - [forum.softuni.bg](http://forum.softuni.bg)



Software  
University

